

Tema 7: Lógica proposicional: Formas normales y cláusulas

José A. Alonso Jiménez
José L. Ruiz Reina

Dpto. de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE SEVILLA

Equivalencia lógica

- Fórmulas equivalentes

- Def.: F y G son equivalentes $\iff \models F \leftrightarrow G$

- Representación: $F \equiv G$

- Ejemplos:

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$

$$p \rightarrow q \equiv \neg p \vee q$$

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

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

- CNS: $F \equiv G \iff$ para toda interpretación I de $\{F, G\}$, $\text{sig}(F, I) = \text{sig}(G, I)$

- Procedimiento de decisión de equivalencias

```
;;; (es-equivalente '(p <-> q) '((p -> q) & (q -> p)))
;;; => T
;;; (es-equivalente '(p -> q) '((- p) / q))
;;; => T
;;; (es-equivalente '(p & q) '(- ((- p) / (- q))))
;;; => T
;;; (es-equivalente '(p / q) '(- ((- p) & (- q))))
;;; => T
(defun es-equivalente (F G)
  (es-valida (equivalencia F G)))
```

Equivalencia lógica

- Propiedades de la equivalencia
 - Si $F \equiv F'$, entonces $\neg F \equiv \neg F'$
 - Si $F \equiv F'$, $G \equiv G'$ y $\star \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$, entonces $F \star G \equiv F' \star G'$
 - Sea G una subfórmula de F y F' la obtenida sustituyendo una ocurrencia de G en F por G' . Si $G \equiv G'$, entonces $F \equiv F'$
- Equivalencias:
 - Idempotencia: $F \vee F \equiv F$, $F \wedge F \equiv F$
 - Conmutatividad: $F \vee G \equiv G \vee F$, $F \wedge G \equiv G \wedge F$
 - Asociatividad: $F \vee (G \vee H) \equiv (F \vee G) \vee H$,
 $F \wedge (G \wedge H) \equiv (F \wedge G) \wedge H$
 - Distributividad: $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$,
 $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$
 - Doble negación: $\neg\neg F \equiv F$
 - Leyes de De Morgan: $\neg(F \wedge G) \equiv \neg F \vee \neg G$,
 $\neg(F \vee G) \equiv \neg F \wedge \neg G$

Forma normal negativa

- Def. de forma normal negativa:
 - Si F es atómica, entonces F y $\neg F$ son formas normales negativas
 - Si F y G son formas normales negativas, entonces $(F \wedge G)$ y $(F \vee G)$ también lo son.
- Ejemplos de formas normales negativas:
 - $(\neg p \vee q) \wedge (\neg q \vee p)$ es forma normal negativa
 - $(p \rightarrow q) \wedge (q \rightarrow p)$ no es forma normal negativa
 - $\neg(p \wedge q)$ no es forma normal negativa
- Transformación a forma normal negativa:
 - Procedimiento FNN(F):
 1. Eliminación de equivalencias
$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$
 2. Eliminación de implicaciones
$$p \rightarrow q \equiv \neg p \vee q$$
 3. Interiorización de negaciones
$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$
$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$
$$\neg\neg p \equiv p$$

Forma normal negativa

- Ejemplos:

$$\text{FNN}(p \leftrightarrow q) = (\neg p \vee q) \wedge (\neg q \vee p)$$

$$\text{FNN}(p \vee \neg q \rightarrow r) = (\neg p \wedge q) \vee r$$

$$\text{FNN}(p \wedge (q \rightarrow r) \rightarrow s) = (\neg p \vee (q \wedge \neg r)) \vee s$$

- Propiedades:

- $\text{FNN}(F)$ es una forma normal negativa

- $\text{FNN}(F) \equiv F$

Forma normal negativa

- Procedimiento de cálculo de forma normal negativa

```
;;; (forma-normal-negativa '((p & (q -> r)) -> s))
;;; => (((- P) / (Q & (- R))) / S)
(defun forma-normal-negativa (F)
  (interioriza-negacion
   (elimina-implicaciones
    (elimina-equivalencias F))))

;;; (elimina-equivalencias '((p <-> q) & (q <-> r)))
;;; => (((P -> Q) & (Q -> P)) & ((Q -> R) & (R -> Q)))
(defun elimina-equivalencias (F)
  (case (tipo F)
    (es-atmica F)
    (es-equivalencia
     (let ((arg1 (elimina-equivalencias (arg1 F)))
           (arg2 (elimina-equivalencias (arg2 F))))
       (conjuncion (implicacion arg1 arg2)
                   (implicacion arg2 arg1))))
    (t (mapcar #'elimina-equivalencias F))))
```

Forma normal negativa

```
;;; > (elimina-implicaciones (elimina-equivalencias '(p <-> q)))
;;; (((- P) / Q) & ((- Q) / P))
(defun elimina-implicaciones (F)
  (case (tipo F)
    (es-atomica F)
    (es-implicacion
     (disyuncion (negacion (elimina-implicaciones (arg1 F)))
                 (elimina-implicaciones (arg2 F))))
    (t (mapcar #'elimina-implicaciones F))))

;;; (interioriza-negacion '(- (- p)))          => P
;;; (interioriza-negacion '(- (p & q)))       => ((- P) / (- Q))
;;; (interioriza-negacion '(- (p / q)))       => ((- P) & (- Q))
;;; (interioriza-negacion '(- (- (p / q))))   => (P / Q)
;;; (interioriza-negacion '(- ((- p) / q)))   => (P & (- Q))
(defun interioriza-negacion (F)
  (case (tipo F)
    (es-atomica F)
    (es-negacion (interioriza-negacion-aux (arg1 F)))
    (t (mapcar #'interioriza-negacion F))))
```

Forma normal negativa

```
(defun interioriza-negacion-aux (F)
  (case (tipo F)
    (es-atmica      (negacion F))
    (es-negacion    (interioriza-negacion (arg1 F)))
    (es-conjuncion  (disyuncion (interioriza-negacion-aux (arg1 F))
                                (interioriza-negacion-aux (arg2 F))))
    (es-disyuncion  (conjuncion (interioriza-negacion-aux (arg1 F))
                                (interioriza-negacion-aux (arg2 F))))
    (t              (negacion F))))
```


Literales

- Literales:
 - F es literal positivo $\iff F$ es fórmula atómica.
 - $\neg F$ es literal negativo $\iff F$ es fórmula atómica
 - F es literal $\iff F$ es literal positivo o negativo
- Procedimiento de reconocimiento de literales

```
;;; (es-literal-positivo 'p)      => T
;;; (es-literal-positivo '(- p)) => NIL
(defun es-literal-positivo (L)
  (es-atomica L))
```

```
;;; (es-literal-negativo '(- p)) => T
;;; (es-literal-negativo 'p)     => NIL
(defun es-literal-negativo (L)
  (and (es-negacion L)
       (es-literal-positivo (arg1 L))))
```

```
;;; (es-literal 'p)      => T
;;; (es-literal '(- p)) => T
(defun es-literal (L)
  (or (es-literal-positivo L)
      (es-literal-negativo L)))
```

- Variables para literales: L, L_1, L_2, \dots

Literales

- Complementario de un literal:

- $\overline{p} = \neg p$

- $\overline{\neg p} = p$

- Procedimiento de cálculo de complementarios

```
;;; (complementario 'p)      => (- P)
;;; (complementario '(- p)) => P
(defun complementario (L)
  (if (es-literal-positivo L)
      (negacion L)
      (second L)))
```

- Procedimiento literales-formula

```
;;; (literales-formula '(p / ((- q) / r)))
;;; => (P (- Q) R)
;;; (literales-formula 'p)
;;; => (P)
;;; (literales-formula '(- p))
;;; => ((- P))
;;; Arg: F es un literal o una disyunción en forma normal
;;;      negativa.
(defun literales-formula (F)
  (if (es-literal F)
      (list F)
      (n-union (literales-formula (arg1 F))
                (literales-formula (arg2 F))
                :test #'equal)))
```

Formas normales conjuntivas

- **Definición de fórmula clausal:**
 - Si F es un literal, entonces F es una fórmula clausal
 - Si F y G son fórmulas clausales, entonces $(F \vee G)$ es un fórmula clausal
- **Ejemplos de fórmulas clausales:**
 - p es una fórmula clausal
 - $\neg p$ es una fórmula clausal
 - $\neg p \vee (q \vee \neg r)$ es una fórmula clausal
 - $\neg p \vee (q \wedge \neg r)$ no es una fórmula clausal
- **Definición de forma normal conjuntiva:**
 - Si F es una fórmula clausal, entonces F es una forma normal conjuntiva
 - Si F y G son formas normales conjuntivas, entonces $(F \wedge G)$ también lo es
- **Ejemplos de formas normales conjuntivas:**
 - $(\neg p \vee q) \wedge (\neg q \vee p)$ es una forma normal conjuntiva
 - $(\neg p \vee q) \wedge (q \rightarrow p)$ no es una forma normal conjuntiva

Formas normales conjuntivas

- Relación entre formas normales:
 - F es forma normal conjuntiva \implies
 $\implies F$ es forma normal negativa
- Transformación a forma normal conjuntiva:
 - Procedimiento $FNC(F)$
 1. Transformación a forma normal negativa
 2. Interiorización de las disyunciones
$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$
$$(p \wedge q) \vee r \equiv (p \vee r) \wedge (q \vee r)$$
 - Ejemplos:
$$FNC(p \wedge (q \rightarrow r)) = p \wedge (\neg q \vee r)$$
$$FNC(\neg(p \wedge (q \rightarrow r))) = (\neg p \vee q) \wedge (\neg p \vee \neg r)$$
- Propiedades:
 - $FNC(F)$ es una forma normal conjuntiva
 - $FNC(F) \equiv F$

Formas normales conjuntivas

- Procedimiento de cálculo de forma normal conjuntiva

```
;;; (forma-normal-conjuntiva '(p & (q -> r)))
;;; => (P & ((- Q) / R))
;;; (forma-normal-conjuntiva '(- (p & (q -> r))))
;;; => (((- P) / Q) & ((- P) / (- R)))
(defun forma-normal-conjuntiva (F)
  (interioriza-disyuncion (forma-normal-negativa F)))

;;; (interioriza-disyuncion '(p / (q & r))) => ((P / Q) & (P / R))
;;; (interioriza-disyuncion '((p & q) / r)) => ((P / R) & (Q / R))
(defun interioriza-disyuncion (F)
  (case (tipo F)
    (es-disyuncion (aplica-distributiva-de-disyuncion
                    (disyuncion (interioriza-disyuncion (arg1 F))
                                (interioriza-disyuncion (arg2 F)))))
    (es-conjuncion (conjuncion (interioriza-disyuncion (arg1 F))
                               (interioriza-disyuncion (arg2 F))))
    (t              F)))
```

Formas normales conjuntivas

```
(defun aplica-distributiva-de-disyuncion (F)
  (cond ((es-conjuncion (arg1 F))
        (conjuncion (interioriza-disyuncion (disyuncion (arg1 (arg1 F))
                                                         (arg2 F)))
                    (interioriza-disyuncion (disyuncion (arg2 (arg1 F))
                                                         (arg2 F))))))
        ((es-conjuncion (arg2 F))
        (conjuncion (interioriza-disyuncion (disyuncion (arg1 F)
                                                         (arg1 (arg2 F))))
                    (interioriza-disyuncion (disyuncion (arg1 F)
                                                         (arg2 (arg2 F)))))))
        (t F)))
```

Transformación a cláusulas

- Cláusula de una fórmula clausal:

- Def.: Sea F una fórmula clausal. Entonces

$$\text{Cláusula}(F) = \begin{cases} \{F\}, & \text{si } F \text{ es un literal;} \\ \text{Cláusula}(F_1) \cup \text{Cláusula}(F_2), & \text{si } F = (F_1 \vee F_2) \end{cases}$$

- Ejemplos:

$$\text{Cláusula}(p) = \{p\}$$

$$\text{Cláusula}(\neg p) = \{\neg p\}$$

$$\text{Cláusula}((\neg p \vee r) \vee (\neg p \vee q)) = \{\neg p, q, r\}$$

- Procedimiento de cálculo

```
;;; (clausula 'p) => (P)
```

```
;;; (clausula '(- p)) => ((- P))
```

```
;;; (clausula '((( - p) / r) / ((- p) / q))) => ((- P) R Q)
```

```
(defun clausula (F)
```

```
  (if (es-literal F)
```

```
      (list F)
```

```
      (n-union (clausula (arg1 F)) (clausula (arg2 F)) :test #'equal)))
```

Transformación a cláusulas

- Cláusulas de una fórmula en forma normal conjuntiva:

- Sea F una fórmula en forma normal conjuntiva.

$$\begin{aligned} \text{Cláusulas-FNC}(F) &= \\ &= \text{Cláusulas-FNC}(F_1) \cup \text{Cláusulas-FNC}(F_2), \\ &\quad \text{si } F = (F_1 \wedge F_2) \\ &= \{\text{Cláusula}(F)\}, \text{ en caso contrario} \end{aligned}$$

- Ejemplos:

$$\text{Cláusulas-FNC}(p \wedge (\neg q \vee r)) = \{\{p\}, \{\neg q, r\}\}$$

$$\text{Cláusulas-FNC}((\neg p \vee q) \wedge (\neg p \vee \neg r)) = \{\{\neg p, q\}, \{\neg p, \neg r\}\}$$

- Procedimiento de cálculo

```
;;; (clausulas-fnc '(p & ((- q) / r)))
;;; => ((P) ((- Q) R))
;;; (clausulas-fnc '((( - p) / q) & ((- p) / (- r))))
;;; => (((- P) Q) ((- P) (- R)))
(defun clausulas-FNC (F)
  (if (es-conjuncion F)
      (n-union (clausulas-FNC (arg1 F))
               (clausulas-FNC (arg2 F))
               :test #'igual-conjunto)
      (list (clausula F))))

;;; (igual-conjunto '((- p) q) '(q (- p))) => T
(defun igual-conjunto (conjunto-1 conjunto-2)
  (and (subsetp conjunto-1 conjunto-2 :test #'equal)
       (subsetp conjunto-2 conjunto-1 :test #'equal)))
```


Transformación a cláusulas

- Cláusulas de una fórmula:

- Def.: $\text{Cláusulas}(F) = \text{Cláusulas-FNC}(\text{FNC}(F))$

- Ejemplos:

- $\text{Cláusulas}(p \wedge (q \rightarrow r)) = \{\{p\}, \{\neg q, r\}\}$

- $\text{Cláusulas}(\neg(p \wedge (q \rightarrow r))) = \{\{\neg p, q\}, \{\neg p, \neg r\}\}$

- Procedimiento de cálculo

- ;;; (clausulas '(p & (q -> r)))

- ;;; => ((P) ((- Q) R))

- ;;; (clausulas '(- (p & (q -> r))))

- ;;; => (((- P) Q) ((- P) (- R)))

- (defun clausulas (F)

- (clausulas-FNC (forma-normal-conjuntiva F)))

Transformación a cláusulas

- Cláusulas de un conjunto de fórmulas:

- Def.: $\text{Cláusulas-conjunto}(S) = \bigcup \{\text{Cláusulas}(F) : F \in S\}$

- Ejemplos:

$\text{Cláusulas-conjunto}(\{p \rightarrow q, q \rightarrow r\}) = \{\{\neg p, q\}, \{\neg q, r\}\}$

$\text{Cláusulas-conjunto}(\{p \rightarrow q, q \leftrightarrow p\}) = \{\{\neg p, q\}, \{\neg q, p\}\}$

- Procedimiento de cálculo

```
;;; > (clausulas-conjunto '(p -> q) (q -> r))
```

```
;;; (((- P) Q) ((- Q) R))
```

```
;;; > (clausulas-conjunto '(p -> q) (q <-> p))
```

```
;;; (((- P) Q) ((- Q) P))
```

```
(defun clausulas-conjunto (S)
```

```
  (union-general (mapcar #'clausulas S)
                 :pred #'igual-conjunto))
```

```
;;; (union-general ()) => NIL
```

```
;;; (union-general '(a)) => (A)
```

```
;;; (union-general '(a) (a b) (b c)) => (A B C)
```

```
(defun union-general (x &key (pred #'equal))
```

```
  (cond ((null x) ())
```

```
        (t (n-union (first x)
```

```
                    (union-general (rest x)
```

```
                    :test pred))))
```

Símbolos proposicionales de cláusulas

- Símbolos proposicionales de un literal:

- Definición:

$$\text{símbolos-proposicionales-literal}(p) = \{p\}$$

$$\text{símbolos-proposicionales-literal}(\neg p) = \{p\}$$

- Procedimiento:

```
;;; (simbolos-proposicionales-literal 'p)      => (P)
;;; (simbolos-proposicionales-literal '(- p)) => (P)
(defun simbolos-proposicionales-literal (L)
  (if (es-literal-positivo L)
      (list L)
      (list (complementario L))))
```

- Símbolos proposicionales de una cláusula:

- Definición:

$$\text{símbolos-proposicionales-cláusula}(C) =$$

$$= \bigcup \{ \text{símbolos-proposicionales-literal}(L) : L \in C \}$$

- Ejemplo:

$$\text{símbolos-proposicionales-cláusula}(\{p, q, \neg p\}) = \{p, q\}$$

- Procedimiento

```
;;; (simbolos-proposicionales-clausula '(p q (- p)))
;;; => (P Q)
(defun simbolos-proposicionales-clausula (C)
  (union-general
   (mapcar #'simbolos-proposicionales-literal C)))
```

Símbolos proposicionales de cláusulas

- Símbolos proposicionales de un conjunto de cláusulas:

- Definición:

$$\begin{aligned} \text{símbolos-proposicionales-conjunto-cláusulas}(S) &= \\ &= \bigcup \{ \text{símbolos-proposicionales-cláusula}(C) : C \in S \} \end{aligned}$$

- Ejemplo:

$$\text{símbolos-proposicionales-conjunto-cláusulas} (\{ \{p, q\}, \{ \neg q, r \} \}) = \{p, q, r\}$$

- Procedimiento

```
;;; (simbolos-proposicionales-conjunto-clausulas '(p q) ((- q) r))  
;;; => (P Q R)  
(defun simbolos-proposicionales-conjunto-clausulas (S)  
  (union-general (mapcar #'simbolos-proposicionales-clausula S)))
```

Interpretaciones de una cláusula

- Interpretaciones de una cláusula:

- Def.: I interpretación de $C \iff I \subseteq \text{símbolos-proposicionales-cláusula}(C)$

- Def.: $\text{interpretaciones-cláusula}(C) = \{I : I \text{ interpretación de } C\}$

- Ej.: $\text{interpretaciones-cláusula}(\{p, q, \neg p\}) = \{\{\}, \{p\}, \{q\}, \{q, p\}\}$

- Procedimiento

```
;;; (interpretaciones-clausula '(p q (- p))) => (NIL (Q) (P) (P Q))
```

```
;;; (interpretaciones-clausula '()) => (NIL)
```

```
(defun interpretaciones-clausula (C)  
  (subconjuntos (simbolos-proposicionales-clausula C)))
```

Interpretaciones de un conjunto de cláusulas

- Interpretaciones de conjuntos de cláusulas:

- Definición:

I interpretación de $S \iff I \subseteq \text{símbolos-proposicionales-conjunto-cláusulas}(S)$

- Def.: $\text{interpretaciones-conjunto-cláusulas}(C) = \{I : I \text{ interpretación de } S\}$

- Ej.: $\text{interpretaciones-conjunto-cláusulas}(\{p, \neg q\}, \{\neg p, q\}) =$
 $= \{\{\}, \{p\}, \{q\}, \{p, q\}\}$

- Procedimiento

```
;;; (interpretaciones-conjunto-clausulas '(p (- q) ((- p) q)))  
;;; => (NIL (Q) (P) (P Q))  
;;; (interpretaciones-conjunto-clausulas '())  
;;; => (NIL)  
(defun interpretaciones-conjunto-clausulas (S)  
  (subconjuntos (simbolos-proposicionales-conjunto-clausulas S)))
```

Modelos de una cláusula

- Modelo de literal:

- Definición:

$$I \models p \iff p \in I$$

$$I \models \neg p \iff p \notin I$$

- Procedimiento

```
;;; (es-modelo-del-literal '(p r) 'p      ) => T
;;; (es-modelo-del-literal '(p r) 'q      ) => NIL
;;; (es-modelo-del-literal '(p r) '(- p)) => NIL
;;; (es-modelo-del-literal '(p r) '(- q)) => T
(defun es-modelo-del-literal (I L)
  (if (es-literal-positivo L)
      (if (member L I) t nil)
      (if (member (complementario L) I) nil t)))
```

Modelos de una cláusula

- Modelo de una cláusula:

- Def.: $I \models C \iff I$ es modelo de algún literal de C

- Ejemplos:

$\{p, r\}$	es modelo de	$\{p, \neg q\}$
$\{r\}$	es modelo de	$\{p, \neg q\}$
$\{q, r\}$	no es modelo de	$\{p, \neg q\}$
$\{p, r\}$	no es modelo de	$\{\}$

- Procedimiento

```
;;; (es-modelo-clausula '(p r) '(p (- q))) => T
;;; (es-modelo-clausula '(r) '(p (- q))) => T
;;; (es-modelo-clausula '(q r) '(p (- q))) => NIL
;;; (es-modelo-clausula '(q r) '()) => NIL
(defun es-modelo-clausula (I C)
  (some #'(lambda (L) (es-modelo-del-literal I L))
        C))
```

- Propiedad (Relación entre modelos de fórmulas clausales y de cláusulas)

- Sea F una fórmula clausal. Entonces

$$I \models F \iff I \models \text{cláusula}(F)$$

Modelos de una cláusula

- Modelos de una cláusula:

- Def.: $\text{Modelos}(C) = \{I \text{ interpretación de } C : I \models C\}$

- Ejemplos:

$\text{Modelos}(\{\neg p, q\}) = \{\{\}, \{q\}, \{p, q\}\}$

$\text{Modelos}(\{\neg p, p\}) = \{\{\}, \{p\}\}$

$\text{Modelos}(\{\}) = \{\}$

- Procedimiento

```
;;; (modelos-clausula '((- p) q)) => (NIL (Q) (P Q))
```

```
;;; (modelos-clausula '((- p) p)) => (NIL (P))
```

```
;;; (modelos-clausula '()) => NIL
```

```
(defun modelos-clausula (C)
```

```
  (remove-if-not #'(lambda (I) (es-modelo-clausula I C))
                 (interpretaciones-clausula C)))
```

- Propiedad (Relación entre modelos de fórmulas clausales y de cláusulas)

- Sea F una fórmula clausal. Entonces

$\text{Modelos}(F) = \text{Modelos}(\text{cláusula}(F))$

Modelos de conjuntos de cláusulas

- Modelo de un conjunto de cláusulas:

- Def.: $I \models S \iff I$ es modelo de todas las cláusulas de S

- Ejemplos:

- $\{p, r\}$ es modelo de $\{\{p, \neg q\}, \{r\}\}$
- $\{p\}$ no es modelo de $\{\{p, \neg q\}, \{r\}\}$
- $\{p, r\}$ es modelo de $\{\}$

- Procedimiento

```
;;; (es-modelo-conjunto-clausulas '(p r) '((p (- q))(r))) => T
;;; (es-modelo-conjunto-clausulas '(p) '((p (- q))(r))) => NIL
;;; (es-modelo-conjunto-clausulas '(p r) '()) => T
(defun es-modelo-conjunto-clausulas (I S)
  (every #'(lambda (C) (es-modelo-clausula I C))
        S))
```

- Propiedad: $I \models F \iff I \models \text{Cláusulas}(F)$

Modelos de conjuntos de cláusulas

- Modelos de un conjunto de cláusulas:

- Def.: $\text{Modelos}(S) = \{I \text{ interpretación de } S : I \models S\}$

- Ejemplos:

$\text{Modelos}(\{\{\neg p, q\}, \{\neg q, p\}\}) = \{\{\}, \{p, q\}\}$

$\text{Modelos}(\{\{\neg p, p\}, \{p\}, \{\neg q\}\}) = \{\}$

$\text{Modelos}(\{\{p, \neg p, q\}\}) = \{\{\}, \{p\}, \{q\}, \{p, q\}\}$

- Procedimiento

```
; (modelos-conjunto-clausulas '(((- p) q) ((- q) p))) => (NIL (P Q))
```

```
; (modelos-conjunto-clausulas '(((- p) q) (p) ((- q)))) => NIL
```

```
; (modelos-conjunto-clausulas '((p (- p) q))) => (NIL (Q) (P) (P Q))
```

```
(defun modelos-conjunto-clausulas (S)
```

```
  (remove-if-not #'(lambda (I) (es-modelo-conjunto-clausulas I S))
```

```
    (interpretaciones-conjunto-clausulas S)))
```

- Propiedad: $\text{Modelos}(F) = \text{Modelos}(\text{Cláusulas}(F))$

Cláusulas válidas y satisfacibles

- Cláusulas válidas:

- Def.: $\models C \iff$ toda interpretación de C es modelo de C

- Ejemplos:

$\{p, q, \neg p\}$ es válida.

$\{p, q, \neg r\}$ no es válida.

- Procedimiento

```
;;; (es-clausula-valida-1 '(p q (- p))) => T
;;; (es-clausula-valida-1 '(p q (- r))) => NIL
;;; (es-clausula-valida-1 '()) => NIL
(defun es-clausula-valida-1 (C)
  (every #'(lambda (I) (es-modelo-clausula I C))
         (interpretaciones-clausula C)))
```

- Propiedad: $\models C \iff C$ contiene un literal y su complementario.

- Procedimiento

```
;;; (es-clausula-valida '(p q (- p))) => T
;;; (es-clausula-valida '(p q (- r))) => NIL
;;; (es-clausula-valida '()) => NIL
(defun es-clausula-valida (C)
  (if (some #'(lambda (L) (member (complementario L)
                                  C
                                  :test #'equal)))
      t))
```

- Propiedad: \emptyset no es válida

Cláusulas válidas y satisfacibles

- Propiedad: Sea F una fórmula clausal. Entonces
 $\models F \iff \models \text{Cláusulas}(F)$

- **Cláusula insatisfacible:**

- Def.: C es insatisfacible $\iff C$ no tiene modelo

- Procedimiento

```
;;; (es-clausula-insatisfacible-1 '(p q (- p))) => NIL
;;; (es-clausula-insatisfacible-1 '(p q (- r))) => NIL
;;; (es-clausula-insatisfacible-1 '()) => T
(defun es-clausula-insatisfacible-1 (C)
  (every #'(lambda (I) (not (es-modelo-clausula I C)))
    (interpretaciones-clausula C)))
```

- Ejemplo: \emptyset es insatisfacible.

- Propiedad: C es insatisfacible $\iff C = \emptyset$

- Procedimiento

```
;;; (es-clausula-insatisfacible '(p q (- p))) => NIL
;;; (es-clausula-insatisfacible '(p q (- r))) => NIL
;;; (es-clausula-insatisfacible '()) => T
(defun es-clausula-insatisfacible (C)
  (null C))
```

Cláusulas válidas y satisfacibles

- **Cláusula satisfacible:**

- Def.: C es satisfacible $\iff C$ tiene modelo

- Ejemplo: $\{p, \neg q\}$ es satisfacible

- **Procedimiento**

```
;;; (es-clausula-satisfacible-1 '(p q (- p))) => T
;;; (es-clausula-satisfacible-1 '(p q (- r))) => T
;;; (es-clausula-satisfacible-1 '()) => NIL
(defun es-clausula-satisfacible-1 (C)
  (some #'(lambda (I) (es-modelo-clausula I C))
        (interpretaciones-clausula C)))
```

- Propiedad: C es satisfacible $\iff C \neq \emptyset$

- **Procedimiento**

```
;;; (es-clausula-satisfacible '(p q (- p))) => T
;;; (es-clausula-satisfacible '(p q (- r))) => T
;;; (es-clausula-satisfacible '()) => NIL
(defun es-clausula-satisfacible (C)
  (not (null C)))
```

Conjuntos de cláusulas válidos y consistentes

- Conjunto válido de cláusulas:

- Def.: $\models S \iff$ todas las interpretaciones de S son modelos de S

- Ejemplos:

- $\{\{\neg p, q\}, \{\neg q, p\}\}$ no es válido

- $\{\{\neg p, p\}, \{\neg q, q\}\}$ es válido

- \emptyset es válido.

- Procedimiento

```
;;; (es-conjunto-valido-de-clausulas-1 '((( - p) q) (( - q) p))) => NIL
;;; (es-conjunto-valido-de-clausulas-1 '((( - p) p) (( - q) q))) => T
;;; (es-conjunto-valido-de-clausulas-1 '()) => T
(defun es-conjunto-valido-de-clausulas-1 (S)
  (every #'(lambda (I) (es-modelo-conjunto-clausulas I S))
    (interpretaciones-conjunto-clausulas S)))
```

Conjuntos de cláusulas válidos y consistentes

- Propiedad: $\models S \iff$ todas las cláusulas de S son válidas

- Procedimiento

```
;;; (es-conjunto-valido-de-clausulas '(((- p) q) ((- q) p))) => NIL
;;; (es-conjunto-valido-de-clausulas '(((- p) p) ((- q) q))) => T
;;; (es-conjunto-valido-de-clausulas '()) => T
(defun es-conjunto-valido-de-clausulas (S)
  (every #'(lambda (C) (es-clausula-valida C))
         S))
```

- Conjunto consistente de cláusulas:

- Def.: S es consistente $\iff S$ tiene modelo
- Def.: S es inconsistente $\iff S$ no tiene modelo

- Ejemplo:

$\{\{\neg p, q\}, \{\neg q, p\}\}$ es consistente
 $\{\{\neg p, q\}, \{p\}, \{\neg q\}\}$ es inconsistente

Conjuntos de cláusulas válidos y consistentes

- Procedimientos

```
;;; (es-conjunto-consistente-de-clausulas '((( - p) q) (( - q) p))) => T
;;; (es-conjunto-consistente-de-clausulas '((( - p) q) (p) (( - q)))) => NIL
(defun es-conjunto-consistente-de-clausulas (S)
  (some #'(lambda (I) (es-modelo-conjunto-clausulas I S))
        (interpretaciones-conjunto-clausulas S)))

;;; (es-conjunto-inconsistente-de-clausulas '((( - p) q) (( - q) p))) => NIL
;;; (es-conjunto-inconsistente-de-clausulas '((( - p) q) (p) (( - q)))) => T
(defun es-conjunto-inconsistente-de-clausulas (S)
  (every #'(lambda (I) (not (es-modelo-conjunto-clausulas I S)))
        (interpretaciones-conjunto-clausulas S)))
```

- Propiedad:

- Sea S un conjunto de fórmulas. Entonces
 S es consistente \iff Cláusulas(S) es consistente

Validez de fórmulas mediante cláusulas

- Decisión de validez por cláusulas (sintáctico)

- Propiedad: $\models F \iff \models \text{Cláusulas}(F)$

- Procedimiento

```
;;; (es-valida-por-clausulas '(p -> p))           => T
;;; (es-valida-por-clausulas '(p -> q))           => NIL
;;; (es-valida-por-clausulas '((p -> q) / (q -> p))) => T
(defun es-valida-por-clausulas (F)
  (es-conjunto-valido-de-clausulas (clausulas F)))
```

Consecuencia lógica mediante cláusulas

- Consecuencia lógica entre cláusulas:

- Def.: $S_1 \models S_2 \iff$ los modelos de S_1 son modelos de S_2

- Ejemplos:

$$\begin{array}{l} \{\{\neg p, q\}, \{\neg q, r\}\} \models \{\{\neg p, r\}\} \\ \{\{p\}\} \not\models \{\{p\}, \{r\}\} \end{array}$$

- Procedimiento

```
; (es-consecuencia-entre-clausulas '(((- p) q)((- q) r)) '(((- p) r))) => T
; (es-consecuencia-entre-clausulas '((p)) '((p) (q))) => NIL
(defun es-consecuencia-entre-clausulas (S1 S2)
  (every #'(lambda (I) (if (es-modelo-conjunto-clausulas I S1)
                          (es-modelo-conjunto-clausulas I S2)
                          t)))
  (interpretaciones-conjunto-clausulas (append S1 S2)))
```

Consecuencia lógica mediante cláusulas

- Propiedad: (Consecuencia entre fórmulas)

- Sea S un conjunto de fórmulas. Entonces,

$$S \models F \iff \text{Cláusulas}(S) \models \text{Cláusulas}(F)$$

$$\iff \text{Cláusulas}(S \cup \{\neg F\}) \text{ es inconsistente}$$

- Procedimientos

```
;;; (es-consecuencia-por-clausulas-1 '((p -> q) (q -> r)) '(p -> r)) => T
```

```
;;; (es-consecuencia-por-clausulas-1 '(p) '(p & q)) => NIL
```

```
(defun es-consecuencia-por-clausulas-1 (S F)
```

```
  (es-consecuencia-entre-clausulas (clausulas-conjunto S)
                                    (clausulas F)))
```

```
;;; (es-consecuencia-por-clausulas '((p -> q) (q -> r)) '(p -> r)) => T
```

```
;;; (es-consecuencia-por-clausulas '(p) '(p & q)) => NIL
```

```
(defun es-consecuencia-por-clausulas (S F)
```

```
  (es-conjunto-inconsistente-de-clausulas
   (clausulas-conjunto (append S (list (negacion F))))))
```

Problema de los animales mediante cláusulas

```
> (setf S '((tiene_pelos / da_leche) -> es_mamifero)
      ((es_mamifero & (tiene_pezuñas / rumia)) -> es_ungulado)
      ((es_ungulado & tiene_cuello_largo) -> es_jirafa)
      ((es_ungulado & tiene_rayas_negras) -> es_cebra)
      (tiene_pelos & (tiene_pezuñas & tiene_rayas_negras)))
      F 'es_cebra)
```

ES_CEBRA

```
> (time (es-consecuencia-por-clausulas S F))
Run time: 1.22 sec.
Space: 894956 Bytes
T
```

Bibliografía

- Alonso Jiménez, J.A. *Lógica computacional* (Univ. de Sevilla, 1997)
(www-cs.us.es/~jalonso/lp/doc/lc-97.ps)
 - Cap. 5 “Equivalencias y formas normales”
 - Cap. 6 “Cláusulas proposicionales”
- Chang, C-L y Lee, R. C-T. *Symbolic Logic and Mechanical Theorem Proving* (Academic Press, 1973)
 - Cap. 2 “The Propositional Logic”.
- Doets, K. *From Logic to Logic Programming* (MIT Press, 1994)
 - Cap. 2: “Propositional Logic”
- Fitting, M. *First-Order Logic and Automated Theorem Proving (2nd, ed.)* (Springer, 1996)
 - Cap. 2: “Propositional Logic”
- Gallier, J.H. *Logic for Computer Science (Foundations of Automatic Theorem Proving)* (Harper & Row, 1986)
 - Cap. 3 “Propositional Logic”

Bibliografía

- Genesereth, M.R. y Nilsson, N.J. *Logical Foundations of Artificial Intelligence* (Morgan Kaufmann, 1987)
 - Cap. 2 “Propositional Logic”
- Paulson, L. *Logic and Proof* (University of Cambridge, 1999)
(www.cl.cam.ac.uk/Teaching/1999/LogProof)
 - Cap. 2: “Propositional logic”
- Schönning, U. *Logic for Computer Scientists*. (Birkhäuser, 1989)
 - Cap. 1 “Propositional Logic”