Capítulo 2

- Lógica proposicional
- Lógica de predicados

Proposición

- Un enunciado es una expresión del lenguaje.
- Una proposición es un enunciado que puede ser calificado o de verdadero (V) o de Falso (F).
- Valores de verdad : L = {V,F} .

Operadores Lógicos

Operador	Símbolo	Compuesta
y	$\wedge$	Conjunción
o (y/o)	$\vee$	Disyunción (Inclusiva)
o (o..o)	Δ	Disyunción exclusiva
entonces	$\rightarrow$	Condicional
si y sólo si	$\leftrightarrow$	Bicondicional
no	$\sim$	Negación

Tablas de los operadores

p	q	$p\wedge q$	$p\vee q$	$p\Delta q$	$p\rightarrow q$	$p\leftrightarrow q$	$\sim p$
V	V	V	V	F	V	V	F
V	F	F	V	V	F	F	F
F	V	F	V	V	V	F	V
F	F	F	F	F	V	V	V

La Condicional

En la condicional $p \rightarrow q$,
p es el antecedente y q es el consecuente.

($p \rightarrow q$, no es conmutativa)

La Condicional

La condicional $p \rightarrow q$ se puede leer:

- p entonces q .
- p sólo si q .
- q si p .

La Condicional

Si la condicional es $p \to q$:

- su recíproca es $q \to p$.
- su contraria es $\sim p \to \sim q$.
- su contrarecíproca es $\sim q \to \sim p$.

Tautología

Una tautología es una proposición compuesta que es **verdadera**, cualesquiera sean los valores de verdad de las proposiciones componentes.

(Una tautología se representa por: **1**)

Ejemplo

$(p \vee \sim p)$ es una tautologia.

Demostración:

p	(p	∨	~p)
0	0	1	1
1	1	1	0

Contradicción

Una contradicción es una proposición compuesta que es **falsa**, cualesquiera sean los valores de verdad de las proposiciones componentes.

(Una contradicción se representa por: **0**)

Ejemplo

$(p \wedge \sim p)$ es una contradicción.

Demostración:

p	(p	∧	~p)
0	0	0	1
1	1	0	0

Proposiciones Equivalentes

Dos proposiciones p y q son equivalentes, si la proposición $p \leftrightarrow q$ es una tautología.

(p es equivalente a q se representa por: $p \equiv q$)

Ejemplo

$(p \to q) \equiv (\sim q \to \sim p)$

Demostración:

p	q	$(p \to q) \leftrightarrow (\sim q \to \sim p)$
0	0	0 1 0 $\boxed{1}$ 1 1 1
0	1	0 1 1 $\boxed{1}$ 0 1 1
1	0	1 0 0 $\boxed{1}$ 1 0 0
1	1	1 1 1 $\boxed{1}$ 0 1 0

Entonces: $(p \to q) \leftrightarrow (\sim q \to \sim p) \equiv 1$

Leyes Lógicas

1) Conmutativa: $p \wedge q \equiv q \wedge p$,
$p \vee q \equiv q \vee p$.
2) Identidad: $p \wedge 1 \equiv p$,
$p \vee 0 \equiv p$.
3) Complemento: $p \wedge \sim p \equiv 0$,
$p \vee \sim p \equiv 1$.
4) Distributiva: $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$,
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$.

Leyes Lógicas +

1) Idempotencia: $p \wedge p \equiv p$, $p \vee p \equiv p$.
2) Acotamiento: $p \wedge 0 \equiv 0$, $p \vee 1 \equiv 1$.
3) Absorción: $p \wedge (p \vee q) \equiv p$,
$p \vee (p \wedge q) \equiv p$.
4) Asociativa: $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$,
$p \vee (q \vee r) \equiv (p \vee q) \vee r$.
5) Involución: $\sim(\sim p) \equiv p$.
6) Opuesto: $\sim 1 \equiv 0$, $\sim 0 \equiv 1$.
7) De Morgan: $\sim(p \wedge q) \equiv \sim p \vee \sim q$,
$\sim(p \vee q) \equiv \sim p \wedge \sim q$.

Leyes Lógicas ++

1) Condicional: $p \to q \equiv \sim p \vee q$.
2) Bicondicional: $p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$.
3) Disyunción Exclusiva: $p \,\Delta\, q \equiv (p \vee q) \wedge \sim(p \wedge q)$.
4) Contraposición: $p \to q \equiv \sim q \to \sim p$.
5) Negación de la Condicional: $\sim(p \to q) \equiv p \wedge \sim q$.
6) Negación de la Bicondicional: $\sim(p \leftrightarrow q) \equiv p \,\Delta\, q$.
7) Absorción generalizada: $p \wedge (\sim p \vee q) \equiv p \wedge q$
$p \vee (\sim p \wedge q) \equiv p \vee q$.

Ejemplo

Simplificar:

$(p \wedge (q \to q)) \wedge (\sim p \to (q \wedge \sim p)) \wedge (r \to p)$

Solución:

$(p \wedge (q \to q)) \wedge (\sim p \to (q \wedge \sim p)) \wedge (r \to p) \equiv$
$(p \wedge (\sim q \vee q)) \wedge (p \vee (q \wedge \sim p)) \wedge (\sim r \vee p) \equiv$
$(p \wedge 1) \wedge (p \vee q) \wedge (\sim r \vee p) \equiv$
$p \wedge (p \vee (q \wedge \sim r)) \equiv$
p

La Implicación

La proposición p implica la proposición q , si la proposición $p \to q$ es una tautología.

(p implica q se representa por: $p \Rightarrow q$)

La Implicación

si p implica q , decimos que :

$\Rightarrow$ p es suficiente para q .

$\Leftarrow$ q es necesario para p .

Ejemplo

$(p \wedge q) \Rightarrow p$

Demostración 1:

p	q	(p	$\wedge$	q)	$\rightarrow$	p
0	0	0	0	0	1	0
0	1	0	0	1	1	0
1	0	1	0	0	1	1
1	1	1	1	1	1	1

Demostración 2:

$$
\begin{aligned}
&(p \wedge q) \rightarrow p \equiv \\
&\sim(p \wedge q) \vee p \equiv \\
&(\sim p \vee \sim q) \vee p \equiv \\
&(\sim p \vee p) \vee \sim q \equiv \\
&1 \vee \sim q \equiv \\
&1
\end{aligned}
$$

Inferencias lógicas

1) Adición: $p \Rightarrow (p \vee q)$.

2) Simplificación: $(p \wedge q) \Rightarrow p$.

3) Modus ponens: $((p \rightarrow q) \wedge p) \Rightarrow q$.

4) Modus tollens: $((p \rightarrow q) \wedge \sim q) \Rightarrow \sim p$.

5) Silogismo disyuntivo: $((p \vee q) \wedge \sim p) \Rightarrow q$.

6) Silogismo hipotético:

$((p \rightarrow q) \wedge (q \rightarrow r)) \Rightarrow (p \rightarrow r)$.

Consecuencia lógica

La proposición q es consecuencia lógica de la proposición p, si $p \Rightarrow q$.

(q es consecuencia lógica de p se representa por: $q \Leftarrow p$)

Ejemplo

$p \Leftarrow (p \wedge q)$

Ya que: $(p \wedge q) \Rightarrow p$

Teorema (Reducción al absurdo)

$q \Leftarrow p$ si y sólo si $(p \wedge \sim q) \equiv 0$.

(La proposición q es consecuencia lógica de la proposiciones p , si y sólo si la proposición $(p \wedge \sim q)$ es una contradicción)

Predicados

Sean $X_1, \ldots, X_n$ conjuntos no vacíos y sea $D \subset X_1 \times \ldots \times X_n$.

Un predicado p definido en D es una función de D en $L = \{V, F\}$.

$(\ p : D \to L\ )$

Predicados

Si p es un predicado en D y $a \in D$, entonces p(a) es una proposición .

Ejemplo

Si $D = Z^+$ y p(x): x es par, entonces:

p(4) es V

p(3) es F

Ejemplo

Si $D = Z \times Z$ y p(x,y): $x < y$, entonces:

p(2,3) es V

p(3,2) es F

Operaciones con predicados

Dados los predicados p y q , se definen los predicados:

1) $(\sim p)(x) \equiv \sim p(x)$.
2) $(p \wedge q)(x) \equiv p(x) \wedge q(x)$.
3) $(p \vee q)(x) \equiv p(x) \vee q(x)$.
4) $(p \Delta q)(x) \equiv p(x) \Delta q(x)$.
5) $(p \to q)(x) \equiv p(x) \to q(x)$.
6) $(p \leftrightarrow q)(x) \equiv p(x) \leftrightarrow q(x)$.

Cuantificador Universal

El cuantificador universal es la expresión "para todo" que se representa por $\forall$.

Si p(x) es un predicado definido en D , la expresión $\forall x \in D : p(x)$, es una proposición.

Cuantificador Existencial

El cuantificador existencial es la expresión "existe" que se representa por $\exists$.

Si p(x) es un predicado definido en D , la expresión $\exists x \in D : p(x)$, es una proposición.

Ejemplo

1) $\forall x \in Z^+: x > 0$, es verdadera

2) $\forall x \in Z: x > 0$, es falsa

3) $\exists x \in Z: x + 2 = 2$, es verdadera

4) $\exists x \in Z^+: x + 2 = 2$, es falsa

Leyes de Predicados

Sean p(x) y q(x) dos predicados:

1) $\sim(\forall x : p(x)) \equiv \exists x : \sim p(x)$.
2) $\sim(\exists x : p(x)) \equiv \forall x : \sim p(x)$.
3) $\forall x : (p(x) \wedge q(x)) \equiv (\forall x : p(x)) \wedge (\forall x : q(x))$.
4) $\exists x : (p(x) \vee q(x)) \equiv (\exists x : p(x)) \vee (\exists x : q(x))$.
5) $\exists x : (p(x) \wedge q(x)) \Rightarrow (\exists x : p(x)) \wedge (\exists x : q(x))$.
6) $\forall x : (p(x) \vee q(x)) \Leftarrow (\forall x : p(x)) \vee (\forall x : q(x))$.

Ejemplo

1) $\sim(\forall x \in Z: x > 0) \equiv \exists x \in Z: x \leq 0$

2) $\sim(\exists x \in Z: x+2 = 2) \equiv \forall x \in Z: x+2 \neq 2$

3) $\sim(\exists x \in Z: x > 0 \wedge x+2 = 2) \equiv \forall x \in Z: x \leq 0 \vee x+2 \neq 2$

4) $\sim(\forall x \in Z: x > 0 \rightarrow x+2 = 2) \equiv \exists x \in Z: x > 0 \wedge x+2 \neq 2$

Teorema (Particularización)

Sea $p : D \rightarrow L$ un predicado y $a \in D$, entonces p(a) es una consecuencia lógica de $(\forall x \in D : p(x))$.

Teorema (Generalización)

Sea $p : D \to L$ un predicado y $a \in D$,
entonces $(\exists x \in D : p(x))$
es una consecuencia lógica de $p(a)$.

Variables ligadas y variables libres

Si $p : D \to L$ es un predicado
en las n variables $X_1, \dots, X_n$, entonces
$\forall X_k : p(X_1, \dots, X_n)$ y
$\exists X_k : p(X_1, \dots, X_n)$
son predicados en las (n-1) variables que no son X_k.

En estos casos se dice que la variable X_k está ligada y que las variables restantes son libres.

Ejemplo

Si $D = Z \times Z$:

1) $(x + y = 0)$ y $(x * y = 0)$, son predicados en las variables x e y.

2) $\forall x$: $x + y = 0$, es un predicado en la variable y, la variable x está ligada y la variable y está libre.

3) $\exists y$: $x + y = 0$, es un predicado en la variable x, la variable y está ligada y la variable x está libre.

4) $\forall x, \exists y$: $x + y = 0$, es una proposición verdadera.

5) $\exists y, \forall x$: $x + y = 0$, es una proposición falsa.

6) $\exists y, \forall x$: $x * y = 0$, es una proposición verdadera.

Números naturales (IN)

Axiomas :

1) IN es un conjunto totalmente ordenado.
2) Existe una función sucesor $S : IN \to IN$ donde $S(n)$ es el número que le sigue a n en el orden total.
3) Existe un único $0 \in IN$ tal que: $0 \notin S(IN)$, donde $S(IN) = \{ S(a) \, / \, a \in IN \}$.

Axioma de inducción

Si $A \subset IN$ es tal que:

1) $0 \in A$.
2) $n \in A \to (n+1) \in A$.

Entonces $A = IN$.

Definiciones por inducción

Si n y $n_0 \in$ IN.

Para definir D(n) $\forall$ n $\geq n_0$:

1) Se define $D(n_0)$.
2) Supuesto definido D(n) $\forall$ n $\geq n_0$, se define D(n+1).

Ejemplo

Definir: n!, $\forall$ n $\geq$ 0.

1) 0! = 1

2) (n+1)! = (n+1).n!, n $\geq$ 0

Ejemplo

Definir: $\sum_{k=1}^{n} a_k \ , \forall \, n \geq 1$

1) $\sum_{k=1}^{1} a_k = a_1$

2) $\sum_{k=1}^{n+1} a_k = \sum_{k=1}^{n} a_k + a_{n+1}$

Demostraciones por inducción

Si n y $n_0 \in$ IN.

Para demostrar P(n) $\forall$ n $\geq n_0$:

1) Se demuestra $P(n_0)$.
2) Supuesto demostrado P(n) $\forall$ n $\geq n_0$, se demuestra P(n+1).

Ejemplo

Demostrar: $\sum_{k=1}^{n} k = \frac{(n)(n+1)}{2}, \forall n \geq 1$

1) $\sum_{k=1}^{1} k = 1 = \frac{(1)(1+1)}{2}$

---

2) $\sum_{k=1}^{n} k = \frac{(n)(n+1)}{2} \quad \rightarrow$

$$\sum_{k=1}^{n} k + (n+1) = \frac{(n)(n+1)}{2} + (n+1) \quad \rightarrow$$

$$\sum_{k=1}^{n+1} k = \frac{(n+1)(n+1+1)}{2}$$

Programa lógico (P.L.)

Un hecho es una proposición no compuesta.
Una regla es una proposición de la forma

$$q \leftarrow (p_1 \wedge \ldots \wedge p_n) .$$

En los hechos y en las reglas las variables son cuantificadas universalmente.

Un programa lógico es un conjunto de hechos y reglas.

Consulta

Una consulta es una proposición.
En las consultas las variables son cuantificadas existencialmente.
Si se ingresa una consulta después de un programa lógico, la salida será "si" en caso de que la consulta sea una consecuencia lógica de programa, y la salida será "no" en caso contrario.
Si la consulta posee variables, la salida indicará todas las soluciones.

Ejemplo

```
/* Hecho: progenitor (x, y) ≡ x es progenitor de y */
  progenitor (josé, roberto) .
  progenitor (rosa, roberto) .
  progenitor (josé, juana) .
  progenitor (rosa, juana) .
  progenitor (roberto, ana) .
  progenitor (ana, jaime) .
```

```
/* Hecho: varón (x) ≡ x es varón */
  varón (josé) .
  varón (roberto) .
  varón (jaime) .
/* Hecho: mujer (x) ≡ x es mujer */
  mujer (rosa).
  mujer (ana).
  mujer (juana).
```

```
/* Consultas y salidas */
  ?: progenitor (josé, juana).
  -: Si .
  ?: progenitor (josé, jaime).
  -: No .
  ?: progenitor (josé, x).
  -: Si .
        x = roberto .
        x = juana .
  ?: progenitor (josé, roberto), varón (josé) .
  -: Si .
```

```
/* Regla: padre (x, y) */
padre (x, y) ← progenitor (x, y), varón (x) .
?: padre (josé, roberto).
-: Si .
?: padre (josé, x).
-: Si .
      x = roberto.
      x = juana.
?: padre (rosa, roberto).
-: No .
```

```
/* más reglas: */

madre (x, y) ←
        progenitor (x, y), mujer (x) .
hijo (x, y)  ←
        progenitor (y, x),
        varón (x) .
hija (x, y)  ←
        progenitor (y, x), mujer (x) .
```

```
abuelo (x, y) ←
        padre (x, z), progenitor (z, y) .
abuela (x, y) ←
        madre (x, z), progenitor (z, y) .
nieto (x, y)  ←
        abuelo (y, x), varón (x) .
nieto (x,y)   ←
        abuela (y, x), varón (x) .
```

```
nieta (x, y)  ←
        abuelo (y, x), mujer (x) .
nieta (x,y)   ←
        abuela (y, x), mujer (x) .
hermanos (x, y) ←
        progenitor (z, x), progenitor (z, y),
        not (misma-persona (x, y)) .
misma-persona (x, x) .
```

```
tío (x, y) ←
    progenitor (z, y), hermanos (x, z), varón (x).
tía (x, y) ←
    progenitor (z, y), hermanos (x, z), mujer (x).
antepasado (x, y) ←
    progenitor (x, z), antepasado (z, y).
antepasado (x, y) ←
        progenitor (x, y).
```

P.L. natural

```
/* natural(x) ≡ x ∈IN */

natural(0).
natural(S(x)) ← natural(x).
```

Ejemplo

```
? natural(3).
  Si.

? natural(0.5).
  No.
```

P.L. igual y diferente

/* igual(x,y) ≡ x = y */

igual(x,x).

/* diferente(x,y) ≡ x ≠ y */

diferente(x,y) ← not(igual(x,y)) .

Ejemplo

? igual(3,3).
Si.
? igual(3,2).
No.
? igual(x,3).
Si
x = 3.
? diferente(3,3).
No.

P.L. menor y menor-igual

/* menor(x,y) ≡ x < y */

menor(S(x),S(y)) ← menor(x,y).
menor(0,S(x)).

/* menor-igual(x,y) ≡ x ≤ y */

menor-igual(S(x),S(y)) ← menor-igual(x,y).
menor-igual(0,x).

Ejemplo

? menor(2,4).
Si.

? menor(2,2).
No.

? menor-igual(2,2).
Si.

P.L. menor-igual2 y diferente2

/* menor-igual2(x,y) ≡ x ≤ y */

menor-igual2(x,y) ← menor(x,y).
menor-igual2(x,y) ← igual(x,y).

/* diferente2(x,y) ≡ x ≠ y */

diferente2(x,y) ← menor(x,y).
diferente2(x,y) ← menor(y,x).

P.L. mayor y mayor-igual

/* mayor(x,y) ≡ x > y */
mayor(x,y) ← menor(y,x).
/* mayor-igual(x,y) ≡ x ≥ y */
mayor-igual(x,y) ← menor-igual(y,x).
/* mayor2(x,y) ≡ x > y */
mayor2(x,y) ← not(menor-igual(x,y)).
/* mayor-igual2(x,y) ≡ x ≥ y */
mayor-igual2(x,y) ← not(menor(x,y)).

P.L. suma y resta

```
/* suma(x,y,z) ≡ z := x + y */

suma(S(x),y,S(z)) ← suma(x,y,z).
suma(0,x,x).

/* resta(x,y,z) ≡ z:= x - y */
resta(x,y,z) ← suma(z,y,x).
```

Ejemplo

```
? suma(2,3,5).
  Si.
? suma(2,3,6).
  No.
? suma(2,3,x).
  Si
  x = 5.
```

```
? suma(x,3,5).
  Si
  x = 2.

? Resta(5,3,x).
  Si
  x = 2.
```

P.L. producto

```
/* producto(x,y,z) ≡ z:= x * y */

producto(S(x),y,z) ← producto(x,y,t),
                     suma(t,y,z).
producto(0,x,0).
```

Ejemplo

```
? producto(2,3,6).
  Si.
? producto(2,3,7).
  No.
? producto(2,3,x).
  Si
  x = 6.
```

P.L. div

```
/* div(x,y,z) ≡ z :=(x DIV y) */

div(x,y,S(z)) ← resta(x,y,t),
                div(t,y,z).
div(x,y,0) ← menor(x,y).
```

Ejemplo

```
? div(17,5,3).
  Si.

? div(17,5,x).
  Si
  x = 3.
```

P.L. mod

```
/* mod(x,y,z)  ≡  z :=(x MOD y)  */

mod(x,y,z) ← resta(x,y,t),
             mod(t,y,z).
mod(x,y,x) ← menor(x,y).
```

Ejemplo

```
? mod(17,5,2).
  Si.

? mod(17,5,x).
  Si
  x = 2.
```

P.L. potencia

/* potencia(x,y,z) ≡ z:= x^y */

```
potencia(x,S(y),z) ← potencia(x,y,t),
                     producto(x,t,z).
potencia(S(x),0,S(0)).
potencia(0,S(y),0).
```

Ejemplo

```
? potencia(2,4,16).
  Si.

? potencia(2,4,x).
  Si
  x = 16.
```

P.L. factorial

```
/* factorial(x,y)  ≡  y:=x!  */

factorial(S(x),y) ← factorial(x,t),
                    producto(S(x),t,y).
factorial(0,S(0)).
```

Ejemplo

? factorial(4,24).
Si.

? factorial(4,x).
Si
x = 24.

Listas

Una lista es la lista vacía o
una lista es una colección ordenada de objetos donde se reconocen dos partes, la cabeza de la lista que es el primer elemento de la lista y la cola de la lista formada por el resto de los elementos.

Listas

Sea L una lista :

1) Si L es la lista vacía,
Representamos L = [ ] .

2) Si L no es la lista vacía,
representamos L = [x | xs] ,
donde x es la cabeza y xs la cola.

P.L. lista

```
lista ( [ ] ).
lista ( [ x | xs] ) ← lista ( xs ).

/*
Una lista es la lista vacía o
una lista es un elemento seguido de una lista
*/
```

P.L. añade

```
/* añade(x,L1,L2) ≡ L2 = [x | L1 ]    */

añade(x,xs,[x | xs]).
```

Ejemplo

? añade(5,[4,7,2],[5,4,7,2]).
Si.

? añade(5,[4,7,2],L).
Si
L = [5,4,7,2].

P.L. pertenece

/* pertenece(x,L) ≡ $x \in L$ */

pertenece(x,[y | ys]) ← pertenece(x,ys).
pertenece(x,[x | xs]) .

Ejemplo

? pertenece(5,[3,7,5,4,2]).
Si.
? pertenece(5,[3,7,4,2]).
No.
? pertenece(x,[3,7,4]).
Si
x = 3
x = 7
x = 4.

P.L. longitud

/* longitud(L,n) ≡ n:= ||L|| ,
donde ||L|| es el número de elementos de L */

longitud([x | xs],S(n)) ← longitud (xs,n).
longitud([ ],0).

Ejemplo

? longitud([2,5,7],3).
Si.

? longitud([2,5,7],x).
Si
x = 3.

P.L. concatena

/* concatena(L1,L2,L3) ≡ L3 = L1L2 */

concatena([x | xs],L,[x | ys]) ← concatena(xs,L,ys).
concatena([ ],L ,L).

Ejemplo

? concatena([2,9,1],[5,2],[2,9,1,5,2]).
Si.

? concatena([2,9,1],[5,2],L).
Si
L = [2,9,1,5,2].

P.L. invierte

/* invierte(L1,L2) ≡ L2 es la lista L1 invertida */

```
invierte([x | xs],L) ← invierte(xs,ts),
                       concatena(ts,[x],L).
invierte([ ],[ ]).
```

Ejemplo

```
? invierte([3,4,7],[7,4,3]).
  Si.

? invierte([3,4,7],L).
  Si
  L = [7,4,3].
```

P.L. posición

/* posición(x,L,n) ≡ n:= la posición del elemento x en la lista L */

```
posición(x,[y | ys],S(n)) ← posición(x,ys,n).
posición(x,[x | xs],S(0)).
```

Ejemplo

```
? posición(5,[7,4,5,9,1],3).
  Si.

? posición(5,[7,4,5,9,1],x).
  Si
  x = 3.
```

```
? posición(5,[7,5,9,1,9,5,1],x).
  Si
  x = 2
  x = 6.

? posición(x,[7,4,5,9],3).
  Si
  x = 5.
```

P.L. menor-elemento

/* menor-elemento(L,x)   x := el menor elemento en la lista L */

```
menor-elemento([x | xs],x) ← menor-elemento(xs,y),
                             menor(x,y).
menor-elemento([x | xs],y) ← menor-elemento(xs,y),
                             not(menor (x,y)).
menor-elemento([ x ],x).
```

Ejemplo

? menor-elemento([5,3,7,2,8],2).
Si.

? menor-elemento([5,3,7,2,8],x).
Si
x = 2.

Matrices

Una matriz de dimensión m x n es una lista, de m listas, de longitud n cada una de ellas.

P.L suma-filas y suma-matrices

```
*/ suma-filas(L1,L2,L3) ≡ L3 := L1 + L2 */
suma-filas([x | xs],[y | ys],[z | zs]) ← suma(x, y, z ),
                                          suma-filas(xs,ys,zs).
suma-filas([ ],[ ],[ ]).
*/ suma-matrices(A,B,C) ≡ C := A + B */
suma-matrices([xs | As],[ys | Bs],[zs | Cs]) ←
                                          suma-filas(xs,ys,zs),
                                          suma-matrices(As,Bs,Cs).
suma-matrices([ ],[ ],[ ]).
```

Ejemplo

? suma-filas([2,5,3],[1,0,4],[3,5,7]).
Si.

? suma-filas([2,5,3],[1,0,4],L).
Si
L = [3,5,7].

? suma-matrices([[4,3],[2,0]],[[1,5],[1,3]],[[5,8],[3,3]]).
Si.

? suma-matrices([[4,3] , [2,0]] , [[1,5] , [1,3]] , L).
Si
L = [[5,8] , [3,3]].

P.L escalarxfila y escalarxmatriz

```
/* escalarxfila(x,L1,L2) ≡ L2 := x.L1 */
escalarxfila(x,[y | ys],[z | zs]) ← producto (x, y, z),
                                      escalarxfila(x,ys,zs).
escalarxfila(x,[ ],[ ]).
/* escalarxmatriz(x,A,B) ≡ B := x.A */
escalarxmatriz(x,[ys|A],[zs|B]) ←
                                      escalarxfila(x,ys,zs ),
                                      escalarxmatriz(x,A,B).
escalarxmatriz(x,[ ],[ ]).
```

Ejemplo

? escalarxfila(4,[3,8,2],[12,32,8]).
Si.

? escalarxfila(4,[3,8,2],L).
Si
L = [12,32,8].

? escalarxmatriz(3, [[4,3] , [2,0]] , [[12,9] , [6,0]]).
Si.

? escalarxmatriz(3, [[4,3] , [2,0]] , L).
Si
L = [[12,9] , [6,0]].