Fórmulas de la Lógica Proposicional

Sesión 2

Edgar Andrade, PhD

Enero de 2019

Departmento de Matemáticas Aplicadas y Ciencias de la Computación

Presentación

En esta sesión estudiaremos:

1. Un poco de historia
2. Fórmulas y la representación del mundo
3. Fórmulas como árboles
4. Funciones recursivas sobre fórmulas

Contenido

Contenido

Lógica

- ☞ La lógica es el estudio de los principios que diferencian los razonamientos válidos de los inválidos.

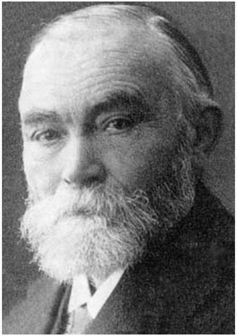
Lógica

- ☞ La lógica es el estudio de los principios que diferencian los razonamientos válidos de los inválidos.
- ☞ Un razonamiento es un discurso que va de unas premisas a una conclusión.

Lógica

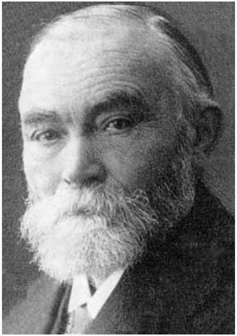
- ☞ La lógica es el estudio de los principios que diferencian los razonamientos válidos de los inválidos.
- ☞ Un razonamiento es un discurso que va de unas premisas a una conclusión.
- ☞ Un razonamiento es válido si no es posible que las premisas sean verdaderas y la conclusión falsa.

Influencias históricas (1/2)



Gotlob Frege (1848–1925)
☞ Las matemáticas se fundamentan en la lógica

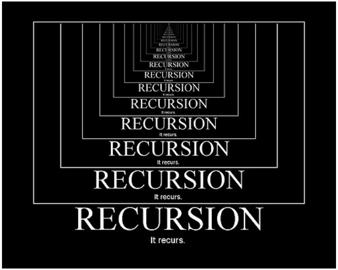
Influencias históricas (1/2)



Gotlob Frege (1848–1925)
☞ Las matemáticas se fundamentan en la lógica

David Hilbert (1862–1943)
☞ Las matemáticas se fundamentan en procedimientos mecánicos

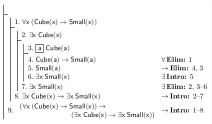
Influencias históricas (2/2)



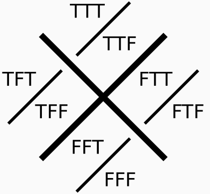
Kurt Gödel (1906–1978)

Sistema lógico

Lenguaje



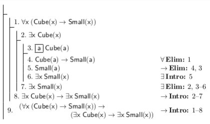
Deducciones



Valores de verdad

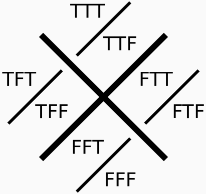
Sistema lógico

Lenguaje



1. $\forall x\,(\mathsf{Cube}(x) \to \mathsf{Small}(x))$
2. $\exists x\,\mathsf{Cube}(x)$
3. $\boxed{a}\ \mathsf{Cube}(a)$
4. $\mathsf{Cube}(a) \to \mathsf{Small}(a)$ — **∀ Elim:** 1
5. $\mathsf{Small}(a)$ — **→ Elim:** 4, 3
6. $\exists x\,\mathsf{Small}(x)$ — **∃ Intro:** 5
7. $\exists x\,\mathsf{Small}(x)$ — **∃ Elim:** 2, 3–6
8. $\exists x\,\mathsf{Cube}(x) \to \exists x\,\mathsf{Small}(x)$ — **→ Intro:** 2–7
9. $(\forall x\,(\mathsf{Cube}(x) \to \mathsf{Small}(x)) \to (\exists x\,\mathsf{Cube}(x) \to \exists x\,\mathsf{Small}(x))$ — **→ Intro:** 1–8

Deducciones



Valores de verdad

Fórmulas

p: El gato está en el árbol.

q: El perro ladra.

$p \wedge q$: El gato está en el árbol y el perro ladra.

Fórmulas

Átomos: p, q, r, $\ldots$

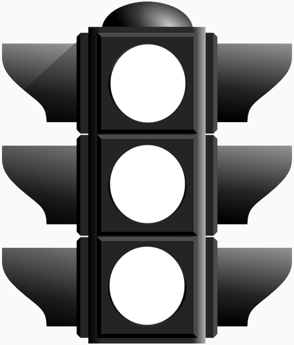
Conectivos lógicos: $\neg$, $\wedge$, $\vee$, $\rightarrow$

Paréntesis: (,)

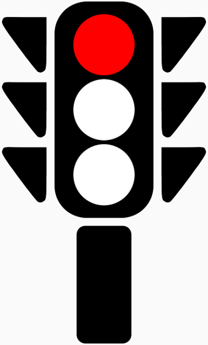
Contenido

Semáforo (1/5)

En un semáforo sólo una luz se prende simultáneamente y siempre hay una luz encendida.

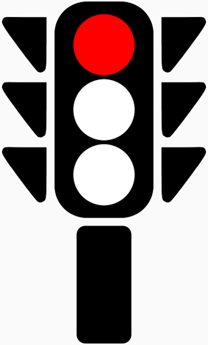


Semáforo (2/5)



p: La luz roja está encendida

Semáforo (2/5)



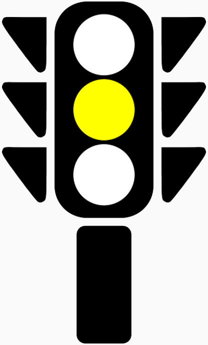
p: La luz roja está encendida

$\neg q$: La luz amarilla no está encendida

$\neg r$: La luz verde no está encendida

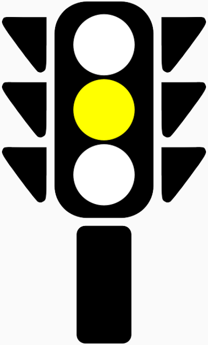
$$p \wedge (\neg q \wedge \neg r)$$

Semáforo (3/5)



q: La luz amarilla está encendida

Semáforo (3/5)



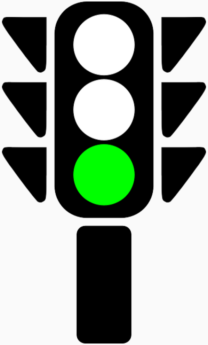
$\neg p$: La luz roja no está encendida

q: La luz amarilla está encendida

$\neg r$: La luz verde no está encendida

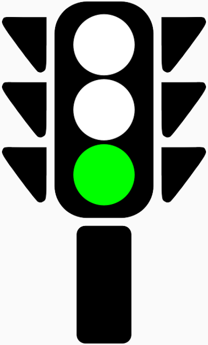
$$q \wedge (\neg p \wedge \neg r)$$

Semáforo (4/5)



q: La luz verde está encendida

Semáforo (4/5)



$\neg p$: La luz roja no está encendida

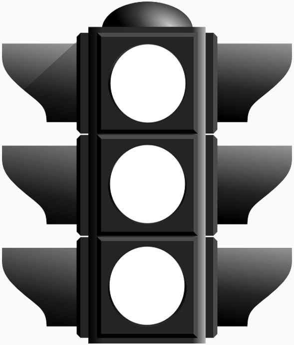
$\neg q$: La luz amarilla no está encendida

q: La luz verde está encendida

$$r \wedge (\neg p \wedge \neg q)$$

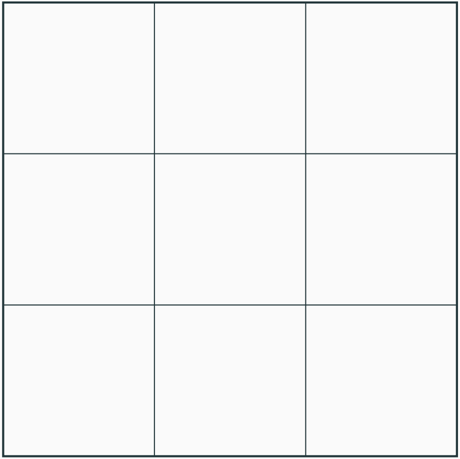
Semáforo (5/5)

En un semáforo sólo una luz se prende simultáneamente y siempre hay una luz encendida.



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

Caballos (1/5)



Poner tres caballos en un tablero 3x3 sin que se ataquen simultáneamente.

Caballos (2/5)

Enumeramos las casillas

1	2	3
4	5	6
7	8	9

Caballos (3/5)

c_1: hay un caballo en 1

c_2: hay un caballo en 2

		
4	5	6
7	8	9

Caballos (3/5)

c_1: hay un caballo en 1

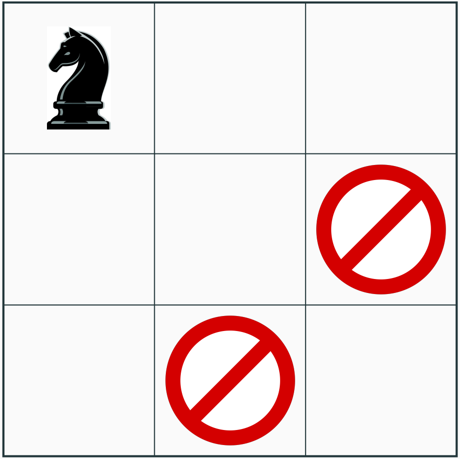
c_2: hay un caballo en 2

$\neg c_3$: no hay un caballo en 3

		
4	5	6
7	8	9

Caballos (4/5)

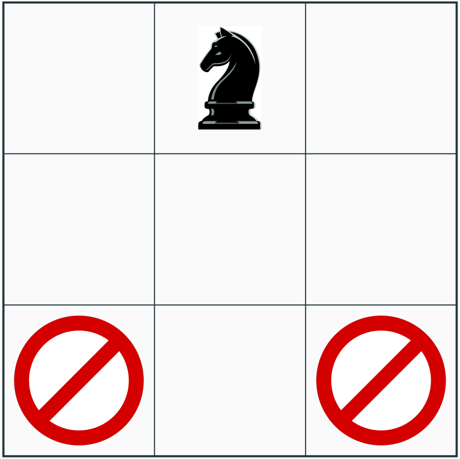
Reglas:
Si hay un caballo en 1, no debe haber un caballo en 6 ni en 8, toda vez que se estarían atacando mútuamente.



Caballos (5/5)

Reglas:

Si hay un caballo en 2, no debe haber un caballo en 7 ni en 9, toda vez que se estarían atacando mútuamente.



Ejercicio (1)

EJERCICIO 1:

Use la lógica proposicional para representar las nueve reglas del problema de los caballos.

Ejercicio (2)

Ejercicio 2:

Use la lógica proposicional para representar el problema de poner 3 torres en un tablero 3x3 sin que se ataquen mútuamente.

Ejercicio (3)

EJERCICIO 3:

Use la lógica proposicional para representar el problema de poner 4 damas en un tablero 4x4 sin que se ataquen mútuamente.

Ejercicio (4)

Ejercicio 4:

Use la lógica proposicional para representar el problema de encontrar un día disponible para hacer una reunión, de acuerdo a las siguientes restricciones:

1. Alejandro sólo tiene disponibilidad para el lunes y el miércoles.
2. Carolina no puede el miércoles.
3. Carlos no puede el viernes.
4. David sólo tiene disponibilidad para el jueves o el viernes.

Contenido

Fórmulas

Definimos un árbol como un objeto con tres atributos

TREE:

.label $\Leftarrow$ Puede ser un átomo o un conectivo

Fórmulas

Definimos un árbol como un objeto con tres atributos

TREE:

.label	$\Leftarrow$ Puede ser un átomo o un conectivo
.left	$\Leftarrow$ Árbol hijo a la izquierda o NULL

Fórmulas

Definimos un árbol como un objeto con tres atributos

TREE:

.label	$\Leftarrow$ Puede ser un átomo o un conectivo
.left	$\Leftarrow$ Árbol hijo a la izquierda o NULL
.right	$\Leftarrow$ Árbol hijo a la derecha o NULL

Ejemplos de fórmulas como árboles (1/2)

$\textsc{Tree}(q, \textsc{null}, \textsc{null})$

q

Ejemplos de fórmulas como árboles (1/2)

$\textsc{Tree}(q, \textsc{null}, \textsc{null})$	q
$\textsc{Tree}(\neg, \textsc{null}, \textsc{Tree}(q, \textsc{null}, \textsc{null}))$	$\begin{array}{c} \neg \\ \vert \\ q \end{array}$

Ejemplos de fórmulas como árboles (1/2)

$\textsc{Tree}(q, \textsc{null}, \textsc{null})$

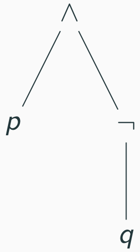
$$q$$

$\textsc{Tree}(\neg, \textsc{null}, \textsc{Tree}(q, \textsc{null}, \textsc{null}))$

$$\begin{array}{c} \neg \\ | \\ q \end{array}$$

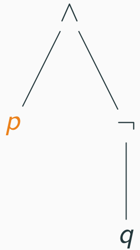
Ejemplos de fórmulas como árboles (2/2)

$$\textsc{Tree}\Big(\wedge,\\ \quad \textsc{Tree}(p, \textsc{null}, \textsc{null}),\\ \quad \textsc{Tree}(\neg, \textsc{null}, \textsc{Tree}(q, \textsc{null}, \textsc{null}))\Big)$$



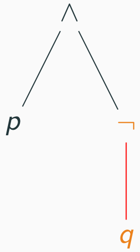
Ejemplos de fórmulas como árboles (2/2)

$$\textsc{Tree}\Big(\wedge,\ \textsc{Tree}(p,\ \textsc{null},\ \textsc{null}),\ \textsc{Tree}(\neg,\ \textsc{null},\ \textsc{Tree}(q,\ \textsc{null},\ \textsc{null}))\Big)$$



Ejemplos de fórmulas como árboles (2/2)

$$\textsc{Tree}\Big(\land,\\ \quad \textsc{Tree}(p, \textsc{null}, \textsc{null}),\\ \quad \textsc{Tree}(\neg, \textsc{null}, \textsc{Tree}(q, \textsc{null}, \textsc{null}))\Big)$$



Contenido

Funciones recursivas (1/3)

Objetivo: Definimos una función que encuentre el conjunto de átomos de una fórmula.

Funciones recursivas (1/3)

Objetivo: Definimos una función que encuentre el conjunto de átomos de una fórmula.

Ejemplos:

Argumento		Resultado
q	$\longmapsto$	$\{q\}$

Funciones recursivas (1/3)

Objetivo: Definimos una función que encuentre el conjunto de átomos de una fórmula.

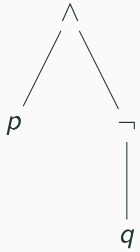
Ejemplos:

Argumento		Resultado
$\neg$		
q	$\longmapsto$	$\{q\}$

Funciones recursivas (1/3)

Objetivo: Definimos una función que encuentre el conjunto de átomos de una fórmula.

Ejemplos:

Argumento		Resultado
	$\longmapsto$	$\{p, q\}$

Funciones recursivas (1/3)

Sea f un árbol arbitrario:

$\textsc{Def Atomos}(f)$:

$\quad \textsc{Si } f.\textsc{right} == \textsc{null}$:

$\quad\quad \textsc{Retornar } \{f.\textsc{label}\}$

$\vdots$

Funciones recursivas (1/3)

Sea f un árbol arbitrario:

$\textsc{Def Atomos}(f)$:

$\quad \textsc{Si}\ f.\textsc{right} == \textsc{null}$:

$\quad\quad \textsc{Retornar}\ \{f.\textsc{label}\}$

$\vdots$

La definición continúa en un momento

Funciones recursivas (1/3)

Sea f un árbol arbitrario:

$\textsc{Def Atomos}(f)$:

  $\textsc{Si}$ $f.\textsc{right} == \textsc{null}$:

    $\textsc{Retornar}$ $\{f.\textsc{label}\}$

$\vdots$

La condición se cumple sii f es un átomo

Funciones recursivas (1/3)

Sea f un árbol arbitrario:

$\textsc{Def Atomos}(f)$:
    $\textsc{Si}$ $f.\textsc{right} == \textsc{null}$:
        $\textsc{Retornar}$ $\{f.\textsc{label}\}$

$\vdots$

Ej:

$f =$Tree(q, null, null)

Funciones recursivas (1/3)

Sea f un árbol arbitrario:

$\textsc{Def Atomos}(f)$:

$\qquad \textsc{Si}\ f.\textsc{right} == \textsc{null}$:

$\qquad\qquad \textsc{Retornar}\ \{f.\textsc{label}\}$

$\vdots$

Ej:

f =Tree(q, null, null)

Funciones recursivas (1/3)

Sea f un árbol arbitrario:

$\textsc{Def Atomos}(f)$:
    $\textsc{Si}\ f.\textsc{right} == \textsc{null}$:
        $\textsc{Retornar}\ \{f.\textsc{label}\}$

$\vdots$

Ej:

f=Tree(q, null, null)

$\textsc{Atomos}(f) = \{f.\text{label}\}$

Funciones recursivas (1/3)

Sea f un árbol arbitrario:

$\textsc{Def Atomos}(f)$:

$\quad \textsc{Si}\ f.\textsc{right} == \textsc{null}$:

$\quad\quad \textsc{Retornar}\ \{f.\textsc{label}\}$

$\vdots$

Ej:

f=Tree(q, null, null)

$\textsc{Atomos}(f)=\{q\}$

Funciones recursivas (1/3)

```
Def Atomos(f):
    Si f.right == null:
        Retornar f.label
```

```
    Si no, si f.label == ¬:
        Retornar Atomos(f.right)
```

⋮

Funciones recursivas (1/3)

```
Def Atomos(f):
    Si f.right == null:
        Retornar f.label
```

Si no, si f.label $== \neg$:

Retornar Atomos(f.right)

$\vdots$

La condición se cumple sii la raíz de f es $\neg$

Funciones recursivas (1/3)

DEF ATOMOS(f):
  SI f.RIGHT == NULL:
    RETORNAR f.LABEL

SI NO, SI f.LABEL == $\neg$:
    RETORNAR ATOMOS(f.RIGHT)

$\vdots$

Ej: f=TREE($\neg$, NULL, Tree(q,null,null))

Funciones recursivas (1/3)

DEF ATOMOS(f):
SI f.RIGHT == NULL:
RETORNAR f.LABEL

SI NO, SI f.LABEL == ¬:
RETORNAR ATOMOS(f.RIGHT)

⋮

Ej: f=TREE(¬, NULL, Tree(q,null,null))

Funciones recursivas (1/3)

DEF ATOMOS(f):
SI f.RIGHT == NULL:
RETORNAR f.LABEL

SI NO, SI f.LABEL == $\neg$:
RETORNAR ATOMOS(f.RIGHT)

$\vdots$

Ej: Observe que f.right=Tree(q,null,null)

Funciones recursivas (1/3)

Def Atomos(f):
  Si f.right == null:
    Retornar f.label

Si no, si f.label == $\neg$:
    Retornar Atomos(f.right)

$\vdots$

Ej: f=Tree($\neg$, null, Tree(q,null,null))

Funciones recursivas (1/3)

Def Atomos(f):
 Si f.right == null:
  Retornar f.label

Si no, si f.label == $\neg$:
  Retornar Atomos(f.right)

$\vdots$

Ej: Luego Atomos(f) = Atomos(f.right)

Funciones recursivas (1/3)

```
Def Atomos(f):
    Si f.right == null:
        Retornar f.label
```

```
    Si no, si f.label == ¬:
            Retornar Atomos(f.right)
```

$\vdots$

Ej: Luego Atomos(f) = Atomos($_{\text{Tree}(q,\text{null},\text{null})}$)

Funciones recursivas (1/3)

Def Atomos(f):
    Si f.right == null:
        Retornar f.label

Si no, si f.label == $\neg$:
        Retornar Atomos(f.right)

$\vdots$

Ej: Luego Atomos(f) $= \{q\}$

Funciones recursivas (1/3)

```
Def Atomos(f):
    Si f.right == null:
        Retornar {f.label}
    Si no, si f.label == ¬:
        Retornar Atomos(f.right)
```

Si no, si f.label $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:

Retornar Atomos(f.left)$\cup$Atomos(f.right)

Funciones recursivas (1/3)

```
DEF ATOMOS(f):
    SI f.RIGHT == NULL:
        RETORNAR {f.LABEL}
    SI NO, SI f.LABEL == ¬:
        RETORNAR ATOMOS(f.RIGHT)
```

SI NO, SI f.LABEL $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:

RETORNAR ATOMOS(f.LEFT)$\cup$ATOMOS(f.RIGHT)

La condición se cumple sii la raíz de f es un conectivo que no es $\neg$

Funciones recursivas (1/3)

DEF ATOMOS(f):
    SI f.RIGHT == NULL:
        RETORNAR $\{f.\text{LABEL}\}$
    SI NO, SI f.LABEL == $\neg$:
        RETORNAR ATOMOS(f.RIGHT)

SI NO, SI f.LABEL $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
        RETORNAR ATOMOS(f.LEFT)$\cup$ATOMOS(f.RIGHT)

Ej: $f =$ Tree$\Big(\wedge,$ Tree(p, null, null), Tree$(\neg$, null, Tree(q, null, null)$)\Big)$

Funciones recursivas (1/3)

$\textsc{Def Atomos}(f)$:
    $\textsc{Si}\ f.\textsc{right} == \textsc{null}$:
        $\textsc{Retornar}\ \{f.\textsc{label}\}$
    $\textsc{Si no, si}\ f.\textsc{label} == \neg$:
        $\textsc{Retornar Atomos}(f.\textsc{right})$

$\textsc{Si no, si}\ f.\textsc{label} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
        $\textsc{Retornar Atomos}(f.\textsc{left}) \cup \textsc{Atomos}(f.\textsc{right})$

Ej: Observe que f.left=Tree(p,null,null)

Funciones recursivas (1/3)

Def Atomos(f):
  Si f.right == null:
    Retornar $\{f.\text{label}\}$
  Si no, si f.label == $\neg$:
    Retornar Atomos(f.right)

  Si no, si $f.\text{label} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
    Retornar Atomos(f.left)$\cup$Atomos(f.right)

Ej: $f =$ Tree$\Big(\wedge$, Tree(p, null, null), Tree($\neg$, null, Tree(q, null, null))$\Big)$

Funciones recursivas (1/3)

Def Atomos(f):
    Si f.right == null:
        Retornar $\{f.\text{label}\}$
    Si no, si f.label == $\neg$:
        Retornar Atomos(f.right)

Si no, si f.label $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
        Retornar Atomos(f.left)$\cup$Atomos(f.right)

Ej: Y que f.right=Tree($\neg$, null, Tree(q, null, null))

Funciones recursivas (1/3)

Def Atomos(f):
Si f.right == null:
Retornar $\{f\text{.label}\}$
Si no, si f.label == $\neg$:
Retornar Atomos(f.right)

Si no, si f.label $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
Retornar Atomos(f.left)$\cup$Atomos(f.right)

Ej: $f =$ Tree$\Big(\wedge,$ Tree(p, null, null), Tree($\neg$, null, Tree(q, null, null))$\Big)$

Funciones recursivas (1/3)

```
Def Atomos(f):
    Si f.right == null:
        Retornar {f.label}
    Si no, si f.label == ¬:
        Retornar Atomos(f.right)
```

$$\text{Si no, si } f.\text{label} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}:$$
$$\text{Retornar Atomos}(f.\text{left}) \cup \text{Atomos}(f.\text{right})$$

Luego $\text{Atomos}(f) = \text{Atomos}(f.\text{left}) \cup \text{Atomos}(f.\text{right})$

Funciones recursivas (1/3)

Def Atomos(f):
  Si f.right == null:
    Retornar {f.label}
  Si no, si f.label == $\neg$:
    Retornar Atomos(f.right)

Si no, si f.label $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
  Retornar Atomos(f.left)$\cup$Atomos(f.right)

Luego Atomos(f)$= \{p\} \cup$Atomos(f.right)

Funciones recursivas (1/3)

```
Def Atomos(f):
    Si f.right == null:
        Retornar {f.label}
    Si no, si f.label == ¬:
        Retornar Atomos(f.right)
```

Si no, si f.label $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:

Retornar Atomos(f.left)$\cup$Atomos(f.right)

Luego Atomos(f)$= \{p\} \cup \{q\}$

Funciones recursivas (1/3)

Def $\textsc{Atomos}(f)$:
  Si $f.\textsc{right} == \textsc{null}$:
    Retornar $\{f.\textsc{label}\}$
  Si no, si $f.\textsc{label} == \neg$:
    Retornar $\textsc{Atomos}(f.\textsc{right})$

  Si no, si $f.\textsc{label} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
    Retornar $\textsc{Atomos}(f.\textsc{left}) \cup \textsc{Atomos}(f.\textsc{right})$

Es decir, Atomos(f)$= \{p, q\}$

Funciones recursivas (1/3)

Función que devuelve el conjunto de átomos de una fórmula f

$\textsc{Def Atomos}(f)$:

  $\textsc{Si}$ $f.\textsc{right} == \textsc{null}$:

    $\textsc{Retornar}$ $\{f.\textsc{label}\}$

  $\textsc{Si no, si}$ $f.\textsc{label} == \neg$:

    $\textsc{Retornar Atomos}(f.\textsc{right})$

  $\textsc{Si no, si}$ $f.\textsc{label} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:

    $\textsc{Retornar Atomos}(f.\textsc{left}) \cup \textsc{Atomos}(f.\textsc{right})$

Funciones recursivas (2/3)

Función que devuelve el conjunto de subfórmulas de una fórmula f

$\textsc{Def Subforms}(f)$:

 $\textsc{Si}$ $f.\textsc{right} == \textsc{null}$:

  $\textsc{Retornar}$ $\{f\}$

 $\textsc{Si no, si}$ $f.\textsc{label} == \neg$:

  $\textsc{Retornar}$ $\{f\} \cup \textsc{Subforms}(f.\textsc{right})$

 $\textsc{Si no, si}$ $f.\textsc{label} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:

  $\textsc{Retornar}$ $\{f\} \cup \textsc{Subforms}(f.\textsc{left}) \cup \textsc{Subforms}(f.\textsc{right})$

Funciones recursivas (2/3)

```
DEF SUBFORMS(f):
    SI f.RIGHT == NULL:
        RETORNAR {f}
    SI NO, SI f.LABEL == ¬:
        RETORNAR {f}∪SUBFORMS(f.RIGHT)
    SI NO, SI f.LABEL ∈ {∧, ∨, →, ↔}:
        RETORNAR {f}∪SUBFORMS(f.LEFT)∪SUBFORMS(f.RIGHT)
```

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

Funciones recursivas (2/3)

Def Subforms(f):
  Si f.right == null:
    Retornar $\{f\}$
  Si no, si f.label == $\neg$:
    Retornar $\{f\} \cup$Subforms(f.right)
  Si no, si f.label $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
    Retornar $\{f\} \cup$Subforms(f.left)$\cup$Subforms(f.right)

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

Subforms(f) $=$

Funciones recursivas (2/3)

```
DEF SUBFORMS(f):
    SI f.RIGHT == NULL:
        RETORNAR {f}
    SI NO, SI f.LABEL == ¬:
        RETORNAR {f}∪SUBFORMS(f.RIGHT)
    SI NO, SI f.LABEL ∈ {∧, ∨, →, ↔}:
        RETORNAR {f}∪SUBFORMS(f.LEFT)∪SUBFORMS(f.RIGHT)
```

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

$= \text{Subforms}\big(\text{Tree}\big(\wedge,\text{Tree}(p,\text{null},\text{null}),\ \text{Tree}(\neg,\text{null},\text{Tree}(q,\text{null},\text{null}))\big)\big)$

Funciones recursivas (2/3)

Def Subforms(f):
Si f.right == null:
Retornar $\{f\}$
Si no, si f.label == $\neg$:
Retornar $\{f\} \cup$ Subforms(f.right)
Si no, si f.label $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
Retornar $\{f\} \cup$ Subforms(f.left) $\cup$ Subforms(f.right)

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

$$= \{f\} \cup \text{Subforms}\big(\text{Tree}(p,\text{null},\text{null})\big) \cup \text{Subforms}\big(\text{Tree}(\neg,\text{null},\text{Tree}(q,\text{null},\text{null}))\big)$$

Funciones recursivas (2/3)

$\textsc{Def Subforms}(f)$:
    $\textsc{Si}$ $f.\textsc{right} == \textsc{null}$:
        $\textsc{Retornar}$ $\{f\}$
    $\textsc{Si no, si}$ $f.\textsc{label} == \neg$:
        $\textsc{Retornar}$ $\{f\} \cup \textsc{Subforms}(f.\textsc{right})$
    $\textsc{Si no, si}$ $f.\textsc{label} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
        $\textsc{Retornar}$ $\{f\} \cup \textsc{Subforms}(f.\textsc{left}) \cup \textsc{Subforms}(f.\textsc{right})$

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

$= \{f\} \cup \{p\} \cup \text{Subforms}\big(\text{Tree}(\neg,\text{null},{\scriptstyle \text{Tree}(q,\text{null},\text{null})})\big)$

Funciones recursivas (2/3)

DEF SUBFORMS(f):
    SI f.RIGHT == NULL:
        RETORNAR $\{f\}$
    SI NO, SI f.LABEL == $\neg$:
        RETORNAR $\{f\}\cup$SUBFORMS(f.RIGHT)
    SI NO, SI f.LABEL $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
        RETORNAR $\{f\}\cup$SUBFORMS(f.LEFT)$\cup$SUBFORMS(f.RIGHT)

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

$= \{f\} \cup \{p\} \cup \{\neg q\} \cup \text{Subforms}(\text{Tree}(q,\text{null},\text{null}))$

Funciones recursivas (2/3)

```
Def Subforms(f):
    Si f.right == null:
        Retornar {f}
    Si no, si f.label == ¬:
        Retornar {f}∪Subforms(f.right)
    Si no, si f.label ∈ {∧, ∨, →, ↔}:
        Retornar {f}∪Subforms(f.left)∪Subforms(f.right)
```

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

$= \{f\} \cup \{p\} \cup \{\neg q\} \cup \{q\}$

Funciones recursivas (2/3)

$\text{DEF SUBFORMS}(f)$:
    $\text{SI } f.\text{RIGHT} == \text{NULL}$:
        $\text{RETORNAR } \{f\}$
    $\text{SI NO, SI } f.\text{LABEL} == \neg$:
        $\text{RETORNAR } \{f\} \cup \text{SUBFORMS}(f.\text{RIGHT})$
    $\text{SI NO, SI } f.\text{LABEL} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
        $\text{RETORNAR } \{f\} \cup \text{SUBFORMS}(f.\text{LEFT}) \cup \text{SUBFORMS}(f.\text{RIGHT})$

Ejemplo: Paso a paso de Subforms(f) para $f = p \wedge \neg q$:

Subforms($p \wedge \neg q$) $= \{p \wedge \neg q, p, \neg q, q\}$

Funciones recursivas (3/3)

Función que sustituye una subfórmula A de una fórmula B por una fórmula A':

$\textsc{Def Sust}[B, A, A']$:

  $\textsc{Si}$ $A \notin \textsc{Subforms}[B]$:

    $\textsc{Retornar}$ B

  $\textsc{Si no, si}$ $B == A$:

    $\textsc{Retornar}$ A'

  $\textsc{Si no, si}$ $B.\textsc{label} == \neg$:

    $\textsc{Retornar Tree}(\neg, \textsc{null}, \textsc{Sust}[B.\textsc{right}, A, A'])$

  $\textsc{Si no, si}$ $B.\textsc{label} \in \{\wedge, \vee \rightarrow, \leftrightarrow\}$:

    $\textsc{Retornar Tree}(B.\textsc{label}, \textsc{Sust}[B.\textsc{left}, A, A'], \textsc{Sust}[B.\textsc{right}, A, A'])$

Funciones recursivas (3/3)

```
DEF SUST[B, A, A′]:
    SI A ∉ SUBFORMS[B]:
        RETORNAR B
    SI NO, SI B == A:
        RETORNAR A′
    SI NO, SI B.LABEL == ¬:
        RETORNAR TREE(¬, NULL, SUST[B.RIGHT, A, A′])
    SI NO, SI B.LABEL ∈ {∧, ∨ →, ↔}:
        RETORNAR TREE(B.LABEL, SUST[B.LEFT, A, A′], SUST[B.RIGHT, A, A′])
```

Ejemplo: Paso a paso de $\text{Sust}\Big[p \wedge \neg q, q, \neg r\Big]$:

Funciones recursivas (3/3)

Def Sust[B, A, A']:
  Si $A \notin$ Subforms[B]:
    Retornar B
  Si no, si $B == A$:
    Retornar A'
  Si no, si B.label $== \neg$:
    Retornar Tree($\neg$, null, Sust[B.right, A, A'])
  Si no, si B.label $\in \{\wedge, \vee \rightarrow, \leftrightarrow\}$:
    Retornar Tree(B.label, Sust[B.left, A, A'], Sust[B.right, A, A'])

Ejemplo: Paso a paso de Sust$\left[p \wedge \neg q, q, \neg r\right]$:

$$\text{Sust}\left[p \wedge \neg q, q, \neg r\right] =$$

Funciones recursivas (3/3)

Def Sust[B, A, A']:
  Si $A \not\in$ Subforms[B]:
    Retornar B
  Si no, si $B == A$:
    Retornar A'
  Si no, si B.label $== \neg$:
    Retornar Tree($\neg$, null, Sust[B.right, A, A'])
  Si no, si B.label $\in \{\wedge, \vee \rightarrow, \leftrightarrow\}$:
    Retornar Tree(B.label, Sust[B.left, A, A'], Sust[B.right, A, A'])

Ejemplo: Paso a paso de $\text{Sust}\Big[p \wedge \neg q, q, \neg r\Big]$:

$= \text{Tree}(B.\text{label}, \text{Sust}[B.\text{left}, A, A'], \text{Sust}[B.\text{right}, A, A'])$

Funciones recursivas (3/3)

$\text{DEF SUST}[B, A, A']$:
    $\text{SI } A \notin \text{SUBFORMS}[B]$:
        $\text{RETORNAR } B$
    $\text{SI NO, SI } B == A$:
        $\text{RETORNAR } A'$
    $\text{SI NO, SI } B.\text{LABEL} == \neg$:
        $\text{RETORNAR TREE}(\neg, \text{NULL}, \text{SUST}[B.\text{RIGHT}, A, A'])$
    $\text{SI NO, SI } B.\text{LABEL} \in \{\wedge, \vee \rightarrow, \leftrightarrow\}$:
        $\text{RETORNAR TREE}(B.\text{LABEL}, \text{SUST}[B.\text{LEFT}, A, A'], \text{SUST}[B.\text{RIGHT}, A, A'])$

Ejemplo: Paso a paso de $\text{Sust}\left[p \wedge \neg q, q, \neg r\right]$:

$= \text{Tree}(\wedge,\ \text{Sust}[_{\text{Tree}(p,\text{null},\text{null})},\ q,\ \neg r],\ \text{Sust}[_{\text{Tree}(\neg,\text{null},\text{Tree}(q,\text{null},\text{null}))},\ q,\ \neg r])$

Funciones recursivas (3/3)

DEF SUST[B, A, A']:
  SI $A \notin$ SUBFORMS[B]:
    RETORNAR B
  SI NO, SI $B == A$:
    RETORNAR A'
  SI NO, SI B.LABEL $== \neg$:
    RETORNAR TREE($\neg$, NULL, SUST[B.RIGHT, A, A'])
  SI NO, SI B.LABEL $\in \{\wedge, \vee \rightarrow, \leftrightarrow\}$:
    RETORNAR TREE(B.LABEL, SUST[B.LEFT, A, A'], SUST[B.RIGHT, A, A'])

Ejemplo: Paso a paso de $\text{Sust}\Big[p \wedge \neg q, q, \neg r\Big]$:

$= \text{Tree}(\wedge,\ \text{Tree}(p,\text{null},\text{null}),\ \text{Sust}[\text{Tree}(\neg,\text{null},\text{Tree}(q,\text{null},\text{null})),\ q,\ \neg r])$

Funciones recursivas (3/3)

```
Def Sust[B, A, A′]:
    Si A ∉ Subforms[B]:
        Retornar B
    Si no, si B == A:
        Retornar A′
    Si no, si B.label == ¬:
        Retornar Tree(¬, null, Sust[B.right, A, A′])
    Si no, si B.label ∈ {∧, ∨ →, ↔}:
        Retornar Tree(B.label, Sust[B.left, A, A′], Sust[B.right, A, A′])
```

Ejemplo: Paso a paso de $\text{Sust}\Big[p \wedge \neg q, q, \neg r\Big]$:

$= \text{Tree}\big(\wedge,\ \text{Tree}(p,\text{null},\text{null}),\ \text{Tree}(\neg,\text{null},\ \text{Sust}[\text{Tree}(q,\text{null},\text{null}),\ q,\ \neg r])\big)$

Funciones recursivas (3/3)

$\textsc{Def Sust}[B, A, A']$:

 $\textsc{Si } A \notin \textsc{Subforms}[B]$:

  $\textsc{Retornar } B$

 $\textsc{Si no, si } B == A$:

  $\textsc{Retornar } A'$

 $\textsc{Si no, si } B.\textsc{label} == \neg$:

  $\textsc{Retornar Tree}(\neg, \textsc{null}, \textsc{Sust}[B.\textsc{right}, A, A'])$

 $\textsc{Si no, si } B.\textsc{label} \in \{\wedge, \vee \rightarrow, \leftrightarrow\}$:

  $\textsc{Retornar Tree}(B.\textsc{label}, \textsc{Sust}[B.\textsc{left}, A, A'], \textsc{Sust}[B.\textsc{right}, A, A'])$

Ejemplo: Paso a paso de $\text{Sust}\Big[p \wedge \neg q, q, \neg r\Big]$:

$= \text{Tree}(\wedge,\ {\scriptstyle \text{Tree}(p,\text{null},\text{null})},\ \text{Tree}(\neg,\text{null},\ \neg r)\big)$

Funciones recursivas (3/3)

Def Sust[B, A, A']:
  Si $A \notin$ Subforms[B]:
    Retornar B
  Si no, si $B == A$:
    Retornar A'
  Si no, si B.label $== \neg$:
    Retornar Tree($\neg$, null, Sust[B.right, A, A'])
  Si no, si B.label $\in \{\wedge, \vee \rightarrow, \leftrightarrow\}$:
    Retornar Tree(B.label, Sust[B.left, A, A'], Sust[B.right, A, A'])

Ejemplo: Paso a paso de $\text{Sust}\Big[p \wedge \neg q, q, \neg r\Big]$:

$$\text{Sust}\Big[p \wedge \neg q, q, \neg r\Big] = p \wedge \neg\neg r$$

Fin de la sesión 2

En esta sesión usted ha aprendido:

1. Un poco de historia
2. Representar situaciones sencillas mediante fórmulas de la lógica proposicional
3. Representar las fórmulas como árboles
4. Encontrar el paso a paso de una función recursiva sobre una fórmula dada