

Programación declarativa (2006–07)

Tema 10: Procesamiento de lenguaje natural

José A. Alonso Jiménez

Andrés Cordon Franco

Grupo de Lógica Computacional

Dpto. Ciencias de la Computación e Inteligencia Artificial

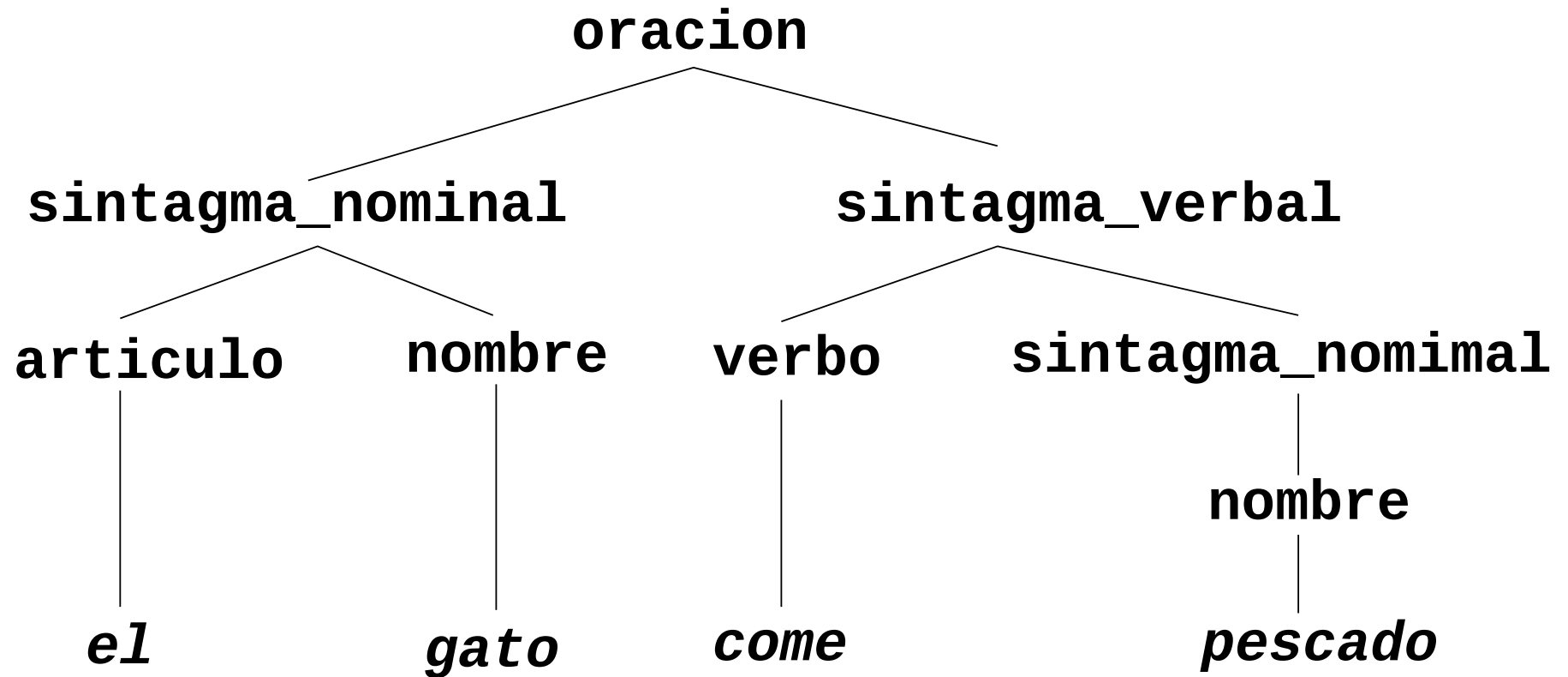
Universidad de Sevilla

Gramáticas libres de contexto: Ejemplo

- Ejemplos de frases:
 - ▶ El gato come pescado
 - ▶ El perro come carne
- Ejemplo de gramática libre de contexto (GLC)
 - oración $\rightarrow$ sintagma_nominal , sintagma_verbal
 - sintagma_nominal $\rightarrow$ nombre
 - sintagma_nominal $\rightarrow$ artículo , nombre
 - sintagma_verbal $\rightarrow$ verbo , sintagma_nominal
 - artículo $\rightarrow$ [el]
 - nombre $\rightarrow$ [gato]
 - nombre $\rightarrow$ [perro]
 - nombre $\rightarrow$ [pescado]
 - nombre $\rightarrow$ [carne]
 - verbo $\rightarrow$ [come]

Gramáticas libres de contexto: Árbol de análisis

- Árbol de análisis



Gramáticas libres de contexto: Definiciones

- Concepto de gramática: $G = (N, T, P, S)$
 - ▶ N : vocabulario no terminal (categorías sintácticas)
 - ▶ T : vocabulario terminal
 - ▶ P : reglas de producción
 - ▶ S : símbolo inicial
- Vocabulario: $V = N \cup T$ es el vocabulario con $N \cap T = \emptyset$
- Derivaciones:
 - ▶ $xAy \Longrightarrow xwy$ mediante $A \Longrightarrow w$
 - ▶ $x \xLongrightarrow{*} y$ si existen $x_1, x_2, \dots, x_n$ tales que
 $x = x_1 \Longrightarrow x_2 \cdots \Longrightarrow x_{n-1} \Longrightarrow x_n = y$
 - ▶ Lenguaje definido por una gramática:
$$L(G) = \{x \in T^* : S \xLongrightarrow{*} x\}$$
 - ▶ Gramáticas libres de contextos (GLC): $A \Longrightarrow w$ con $A \in N$ y $w \in V^*$

Reconocedor de GLC mediante append

- Representación de oraciones en Prolog
[el , gato , come , pescado]
[el , perro , come , carne]
- Reconocedor de GLC en Prolog mediante append:
 - ▶ Sesión (con coste)
?— **time**(oración([el , gato , come , pescado])).
% 178 inferences in 0.00 seconds (Infinite Lips)
Yes
?— **time**(oración([el , come , pescado])).
% 349 inferences in 0.00 seconds (Infinite Lips)
No

- Definición

```
oración(O)      :- sintagma_nominal(SN) ,
                  sintagma_verbal(SV) ,
                  append(SN,SV,O) .
```

sintagma_nominal(SN) :- nombre(SN).

```
sintagma_nominal(SN) :- articulo(A),
                        nombre(N),
                        append(A,N,SN).
```

```
sintagma_verbal(SV) :- verbo(V),
                        sintagma_nominal(SN),
                        append(V, SN, SV).
```

artículo ([el]).

nombre ([gato]). nombre ([perro]).

nombre ([pescado]). nombre ([carne]).

```
verbo ([ come ]).
```

Reconocedor de GLC mediante append

- Otros usos de la gramática

- ▶ Generación de las oraciones

?– oración(O).

O = [gato , come , gato] ; O = [gato , come , perro]

?– **findall**(_O, oración(_O), _L), **length**(_L, N).

N = 64

- ▶ Reconocedor de las categorías gramaticales

?– sintagma_nominal([el , gato]).

Yes

?– sintagma_nominal([un , gato]).

No

- ▶ Generador de las categorías gramaticales

?– **findall**(_SN, sintagma_nominal(_SN), L).

L = [[gato] , [perro] , [pescado] , [carne] , [el , gato] ,
[el , perro] , [el , pescado] , [el , carne]]

Listas de diferencias

- Representaciones de $[a, b, c]$ como listas de diferencias:

$[a, b, c, d] - [d]$

$[a, b, c, 1, 2, 3] - [1, 2, 3]$

$[a, b, c | X] - X$

$[a, b, c] - []$

- Concatenación de listas de diferencias:

- ▶ Programa:

`conc_Id(A-B, B-C, A-C).`

- ▶ Sesión:

`?- conc_Id([a, b | RX]-RX, [c, d | RY]-RY, Z-[]).`

`RX = [c, d]      RY = []      Z = [a, b, c, d]`

`?- conc_Id([a, b |_RX]-_RX, [c, d |_RY]-_RY, Z-[]).`

`Z = [a, b, c, d]`

Yes

Reconocedor de GLC mediante listas de diferencia

- Sesión (y ganancia en eficiencia)

?— **time**(oración([el , gato , come , pescado] — [])).

% 9 inferences in 0.00 seconds (Infinite Lips)

Yes

?— **time**(oración([el , come , pescado] — [])).

% 5 inferences in 0.00 seconds (Infinite Lips)

No

- Definición

```
oración(A-B)      :- sintagma_nominal(A-C),
                    sintagma_verbal(C-B).
```

sintagma_nominal(A-B) :- nombre(A-B).

$$\text{sintagma_nominal}(A-B) \text{ :- artículo}(A-C), \text{ nombre}(C-B).$$

```
sintagma_verbal(A-B) :- verbo(A-C),
                           sintagma_nominal(C-B).
```

artículo ([el |A]—A).

nombre ([gato | A]—A).

nombre ([perro | A] - A).

nombre ([pescado | A] – A) .

nombre ([carne | A] – A).

verbo ([come | A] – A).

Reconocedor de GLC mediante listas de diferencia

- Otros usos de la gramática

- ▶ Generación de las oraciones

?– oración(O–[]).

O = [gato , come, gato] ; O = [gato , come, perro]

?– **findall**(_O, oración(_O–[]), _L), **length**(_L, N).

N = 64

- ▶ Reconocedor de las categorías gramaticales

?– sintagma_nominal([el , gato] –[]).

Yes

?– sintagma_nominal([un , gato] –[]).

No

- ▶ Generador de las categorías gramaticales

?– **findall**(_SN, sintagma_nominal(_SN–[]), L).

L = [[gato] , [perro] , [pescado] , [carne] , [el , gato] ,
[el , perro] , [el , pescado] , [el , carne]]

Gramáticas de cláusulas definidas: Ejemplo

- Ejemplo de GCD

- ▶ Definición

oración $\longrightarrow$ sintagma_nominal, sintagma_verbal.

sintagma_nominal $\longrightarrow$ nombre.

sintagma_nominal $\longrightarrow$ artículo, nombre.

sintagma_verbal $\longrightarrow$ verbo, sintagma_nominal.

artículo $\longrightarrow$ [el].

nombre $\longrightarrow$ [gato].

nombre $\longrightarrow$ [perro].

nombre $\longrightarrow$ [pescado].

nombre $\longrightarrow$ [carne].

verbo $\longrightarrow$ [come].

Gramáticas de cláusulas definidas: Usos

- Reconocimiento de oraciones

?— oración ([el , gato , come , pescado] , []).

Yes

?— oración ([el , come , pescado] , []).

No

- Generación de las oraciones

?— oración(O, []).

O = [gato , come , gato] ;

O = [gato , come , perro] ;

O = [gato , come , pescado]

Yes

?— **findall**(_O, oración(_O, []) , _L) , **length**(_L, N).

N = 64

Gramáticas de cláusulas definidas: Usos

- Reconocedor de las categorías gramaticales

?— `sintagma_nominal([el , gato] , [])`.

Yes

?— `sintagma_nominal([un , gato] , [])`.

No

- Generador de las categorías gramaticales

?— `findall(_SN, sintagma_nominal(_SN, []) , L)`.

`L = [[gato] , [perro] , [pescado] , [carne] ,
[el , gato] , [el , perro] , [el , pescado] , [el , carne]]`

- Determinacion de elementos

?— `oración([X, gato , Y, pescado] , [])`.

`X = el`

`Y = come ;`

No

Gramáticas de cláusulas definidas: Usos

- La relación `phrase`

?— `phrase(oración , [ el , gato , come , pescado ] ) .`

Yes

?— `phrase(sintagma_nominal , L ) .`

`L = [ gato ] ;`

`L = [ perro ]`

Yes

- Compilación

```
?— listing ([ oración , sintagma_nominal ,  
             sintagma_verbal , artículo , nombre , verbo ] ).
```

```
oración(A, B)      :- sintagma_nominal(A, C),
                    sintagma_verbal(C, B).
```

```
sintagma_nominal(A, B) :- nombre(A, B).
```

```
sintagma_nominal(A, B) :- artículo(A, C),
                             nombre(C, B).
```

```
sintagma_verbal(A, B) :- verbo(A, C),
                           sintagma_nominal(C, B).
```

artículo $([e|A], A)$.

```
nombre ([ gato | A], A).
```

nombre ([perro | A] , A).

```
nombre ([pescado | A], A).
```

nombre ([carne | A], A).

```
verbo ([ come | A ], A).
```

Reglas recursivas en GCD: Primera propuesta

- Problema: Extender el ejemplo de GCD para aceptar oraciones como [el ,gato ,come ,pescado ,o ,el ,perro ,come ,pescado]

- Primera propuesta

oración $\longrightarrow$ oración , conjunción , oración .

oración $\longrightarrow$ sintagma_nominal , sintagma_verbal .

sintagma_nominal $\longrightarrow$ nombre .

sintagma_nominal $\longrightarrow$ artículo , nombre .

sintagma_verbal $\longrightarrow$ verbo , sintagma_nominal .

artículo $\longrightarrow$ [el] .

nombre $\longrightarrow$ [gato] .

nombre $\longrightarrow$ [perro] .

nombre $\longrightarrow$ [pescado] .

nombre $\longrightarrow$ [carne] .

verbo $\longrightarrow$ [come] .

conjunción $\longrightarrow$ [y] .

conjunción $\longrightarrow$ [o] .

Reglas recursivas en GCD: Primera propuesta

- Sesión

```
?— oración ([ el , gato , come , pescado , o ,  
            el , perro , come , pescado ] , []).
```

ERROR: Out of local stack

```
?— listing (oración).
```

```
oración(A, B) :—  
    oración(A, C) ,  
    conjunción(C, D) ,  
    oración(D, B).
```

```
oración(A, B) :—  
    sintagma_nominal(A, C) ,  
    sintagma_verbal(C, B).
```

Yes

Reglas recursivas en GCD: Segunda propuesta

- Segunda propuesta

oración $\longrightarrow$ sintagma_nominal , sintagma_verbal .

oración $\longrightarrow$ oración , conjunción , oración .

sintagma_nominal $\longrightarrow$ nombre .

sintagma_nominal $\longrightarrow$ artículo , nombre .

sintagma_verbal $\longrightarrow$ verbo , sintagma_nominal .

artículo $\longrightarrow$ [el] .

nombre $\longrightarrow$ [gato] .

nombre $\longrightarrow$ [perro] .

nombre $\longrightarrow$ [pescado] .

nombre $\longrightarrow$ [carne] .

verbo $\longrightarrow$ [come] .

conjunción $\longrightarrow$ [y] .

conjunción $\longrightarrow$ [o] .

Reglas recursivas en GCD: Segunda propuesta

- Sesión

?— oración ([el , gato , come , pescado , o ,
el , perro , come , pescado] , []) .

Yes

?— oración ([un , gato , come] , []) .

ERROR: Out of local stack

Reglas recursivas en GCD: Tercera propuesta

- Tercera propuesta

oración $\longrightarrow$ oración_simple .

oración $\longrightarrow$ oración_simple , conjunción , oración .

oración_simple $\longrightarrow$ sintagma_nominal ,
sintagma_verbal .

sintagma_nominal $\longrightarrow$ nombre .

sintagma_nominal $\longrightarrow$ artículo , nombre .

sintagma_verbal $\longrightarrow$ verbo , sintagma_nominal .

artículo $\longrightarrow$ [el] .

nombre $\longrightarrow$ [gato] .

nombre $\longrightarrow$ [perro] .

nombre $\longrightarrow$ [pescado] .

nombre $\longrightarrow$ [carne] .

verbo $\longrightarrow$ [come] .

conjunción $\longrightarrow$ [y] .

conjunción $\longrightarrow$ [o] .

Reglas recursivas en GCD: Tercera propuesta

- Sesión

?— oración ([el , gato , come , pescado ,
o , el , perro , come , pescado] , []).

Yes

?— oración ([un , gato , come] , []).

No

GCD para un lenguaje formal

- GCD para el lenguaje formal $\{a^n b^n : n \in \mathbb{N}\}$

- ▶ Sesión

?- s([a,a,b,b],[]).

Yes

?- s([a,a,b,b,b],[]).

No

?- s(X,[]).

X = [] ;

X = [a, b] ;

X = [a, a, b, b]

Yes

- ▶ GCD

s $\longrightarrow$ [].

s $\longrightarrow$ i, s, d.

i $\longrightarrow$ [a].

d $\longrightarrow$ [b].

Árbol de análisis con GCD

?— oración(A, [el , gato , come , pescado] , []).

A = o(sn(art(el), n(gato)), sv(v(come), sn(n(pescado))))

Definición

oración(o(SN, SV)) $\longrightarrow$ sintagma_nominal(SN),
sintagma_verbal(SV).

sintagma_nominal(sn(N)) $\longrightarrow$ nombre(N).

sintagma_nominal(sn(Art, N)) $\longrightarrow$ artículo(Art),
nombre(N).

sintagma_verbal(sv(V, SN)) $\longrightarrow$ verbo(V),
sintagma_nominal(SN).

artículo(art(el)) $\longrightarrow$ [el].

nombre(n(gato)) $\longrightarrow$ [gato].

nombre(n(perro)) $\longrightarrow$ [perro].

nombre(n(pescado)) $\longrightarrow$ [pescado].

nombre(n(carne)) $\longrightarrow$ [carne].

verbo(v(come)) $\longrightarrow$ [come].

Árbol de análisis con GCD

- Compilación

?— **listing** ([oración , sintagma_nominal , nombre]).

oración(o(A,B) ,C,D) :- sintagma_nominal(A,C,E) ,
sintagma_verbal(B,E,D).

sintagma_nominal(sn(A) ,B,C) :- nombre(A,B,C).

sintagma_nominal(sn(A,B) ,C,D) :- artículo(A,C,E) ,
nombre(B,E,D).

nombre(n(gato) ,[gato |A] ,A).

nombre(n(perro) ,[perro |A] ,A).

nombre(n(pescado) ,[pescado |A] ,A).

nombre(n(carne) ,[carne |A] ,A).

Concordancia de género en GCD

- Sesión

?— oración ([el , gato , come , pescado] , []).

Yes

?— oración ([la , gato , come , pescado] , []).

No

?— oración ([la , gata , come , pescado] , []).

Yes

Concordancia de género en GCD

- Definición

oración $\longrightarrow$ sintagma_nominal, sintagma_verbal.

sintagma_nominal $\longrightarrow$ nombre(_).

sintagma_nominal $\longrightarrow$ artículo(G), nombre(G).

sintagma_verbal $\longrightarrow$ verbo, sintagma_nominal.

artículo(masculino) $\longrightarrow$ [el].

artículo(femenino) $\longrightarrow$ [la].

nombre(masculino) $\longrightarrow$ [gato].

nombre(femenino) $\longrightarrow$ [gata].

nombre(masculino) $\longrightarrow$ [pescado].

verbo $\longrightarrow$ [come].

Concordancia de número en GCD

- Sesión

?— oración ([el , gato , come , pescado] , []).

Yes

?— oración ([los , gato , come , pescado] , []).

No

?— oración ([los , gatos , comen , pescado] , []).

Yes

Concordancia de número en GCD

- Definición

oración $\longrightarrow$ sintagma_nominal(N), sintagma_verbal(N).

sintagma_nominal(N) $\longrightarrow$ nombre(N).

sintagma_nominal(N) $\longrightarrow$ artículo(N), nombre(N).

sintagma_verbal(N) $\longrightarrow$ verbo(N), sintagma_nominal(_).

artículo(singular) $\longrightarrow$ [el].

artículo(plural) $\longrightarrow$ [los].

nombre(singular) $\longrightarrow$ [gato].

nombre(plural) $\longrightarrow$ [gatos].

nombre(singular) $\longrightarrow$ [perro].

nombre(plural) $\longrightarrow$ [perros].

nombre(singular) $\longrightarrow$ [pescado].

nombre(singular) $\longrightarrow$ [carne].

verbo(singular) $\longrightarrow$ [come].

verbo(plural) $\longrightarrow$ [comen].

Ejemplo de GCD no GCL

- GCD para el lenguaje formal $\{a^n b^n c^n : n \in \mathbb{N}\}$

- ▶ Sesión

?- s([a,a,b,b,c,c],[]).

Yes

?- s([a,a,b,b,b,c,c],[]).

No

?- s(X,[]).

X = [] ;

X = [a, b, c] ;

X = [a, a, b, b, c, c]

Yes

Ejemplo de GCD no GCL

- GCD para el lenguaje formal $\{a^n b^n c^n : n \in \mathbb{N}\}$

- ▶ GCD

$s \longrightarrow \text{bloque_a}(N), \text{bloque_b}(N), \text{bloque_c}(N).$

$\text{bloque_a}(0) \longrightarrow [].$

$\text{bloque_a}(\text{suc}(N)) \longrightarrow [a], \text{bloque_a}(N).$

$\text{bloque_b}(0) \longrightarrow [].$

$\text{bloque_b}(\text{suc}(N)) \longrightarrow [b], \text{bloque_b}(N).$

$\text{bloque_c}(0) \longrightarrow [].$

$\text{bloque_c}(\text{suc}(N)) \longrightarrow [c], \text{bloque_c}(N).$

GCD con llamadas a Prolog

- GCD para el lenguaje formal $L = \{a^{2n}b^{2n}c^{2n} : n \in \mathbb{N}\}$

- ▶ Ejemplos

?- s([a,a,b,b,c,c],[]).

Yes

?- s([a,b,c],[]).

No

?- s(X,[]).

X = [] ;

X = [a,a,b,b,c,c] ;

X = [a,a,a,a,b,b,b,b,c,c,c,c] ;

X = [a,a,a,a,a,a,b,b,b,b,b,b,c,c,c,c,c,c] ;

Yes

GCD con llamadas a Prolog

- GCD para el lenguaje formal $L = \{a^{2n}b^{2n}c^{2n} : n \in \mathbb{N}\}$

- ▶ GCD

$s \longrightarrow \text{bloque_a}(N), \text{bloque_b}(N), \text{bloque_c}(N),$
 $\{\text{par}(N)\}.$

$\text{bloque_a}(0) \longrightarrow [].$

$\text{bloque_a}(s(N)) \longrightarrow [a], \text{bloque_a}(N).$

$\text{bloque_b}(0) \longrightarrow [].$

$\text{bloque_b}(s(N)) \longrightarrow [b], \text{bloque_b}(N).$

$\text{bloque_c}(0) \longrightarrow [].$

$\text{bloque_c}(s(N)) \longrightarrow [c], \text{bloque_c}(N).$

$\text{par}(0).$

$\text{par}(s(s(N))) \text{ :- } \text{par}(N).$

Separación de reglas y lexicón

- Sesión

?— oración ([el , gato , come , pescado] , []) .

Yes

?— oración ([el , come , pescado] , []) .

No

- Lexicón

lex (el , artículo) .

lex (gato , nombre) .

lex (perro , nombre) .

lex (pescado , nombre) .

lex (carne , nombre) .

lex (come , verbo) .

Separación de reglas y lexicón

- Regla

oración $\longrightarrow$ sintagma_nominal , sintagma_verbal .

sintagma_nominal $\longrightarrow$ nombre .

sintagma_nominal $\longrightarrow$ artículo , nombre .

sintagma_verbal $\longrightarrow$ verbo , sintagma_nominal .

artículo $\longrightarrow$ [Palabra] , { lex (Palabra , artículo) } .

nombre $\longrightarrow$ [Palabra] , { lex (Palabra , nombre) } .

verbo $\longrightarrow$ [Palabra] , { lex (Palabra , verbo) } .

Separación de reglas y lexicón con concordancia

?— oración ([el , gato , come , pescado] , []). ==> Yes

?— oración ([los , gato , come , pescado] , []). ==> No

?— oración ([los , gatos , comen , pescado] , []). ==> Yes

- Lexicón

lex (el , artículo , singular).

lex (los , artículo , plural).

lex (gato , nombre , singular).

lex (gatos , nombre , plural).

lex (perro , nombre , singular).

lex (perros , nombre , plural).

lex (pescado , nombre , singular).

lex (pescados , nombre , plural).

lex (carne , nombre , singular).

lex (carnes , nombre , plural).

lex (come , verbo , singular).

lex (comen , verbo , plural).

Separación de reglas y lexicón con concordancia

- Reglas

oración $\longrightarrow$ sintagma_nominal(N) ,
sintagma_verbal(N) .

sintagma_nominal(N) $\longrightarrow$ nombre(N) .

sintagma_nominal(N) $\longrightarrow$ artículo (N) , nombre(N) .

sintagma_verbal(N) $\longrightarrow$ verbo(N) ,
sintagma_nominal(_)

artículo (N) $\longrightarrow$ [Palabra] , { lex (Palabra , artículo , N) } .

nombre(N) $\longrightarrow$ [Palabra] , { lex (Palabra , nombre , N) } .

verbo(N) $\longrightarrow$ [Palabra] , { lex (Palabra , verbo , N) } .

Lexicón con género y número

- Sesión

?— oración ([la , profesora , lee , un , libro] , []).

Yes

?— oración ([la , profesor , lee , un , libro] , []).

No

?— oración ([los , profesores , leen , un , libro] , []).

Yes

?— oración ([los , profesores , leen] , []).

Yes

?— oración ([los , profesores , leen , libros] , []).

Yes

Lexicón con género y número

- Lexicón

lex (el , determinante , masculino , singular).

lex (los , determinante , masculino , plural).

lex (la , determinante , femenino , singular).

lex (las , determinante , femenino , plural).

lex (un , determinante , masculino , singular).

lex (una , determinante , femenino , singular).

lex (unos , determinante , masculino , plural).

lex (unas , determinante , femenino , plural).

lex (profesor , nombre , masculino , singular).

lex (profesores , nombre , masculino , plural).

lex (profesora , nombre , femenino , singular).

lex (profesoras , nombre , femenino , plural).

lex (libro , nombre , masculino , singular).

lex (libros , nombre , masculino , plural).

lex (lee , verbo , singular).

Lexicón con género y número

- Reglas

oración $\longrightarrow$ sintagma_nominal(N) ,
verbo(N) ,
complemento .

complemento $\longrightarrow$ [] .

complemento $\longrightarrow$ sintagma_nominal(_).

sintagma_nominal(N) $\longrightarrow$ nombre(_,N).

sintagma_nominal(N) $\longrightarrow$ determinante(G,N) ,
nombre(G,N).

determinante(G,N) $\longrightarrow$ [P] , { lex (P, determinante ,G,N) } .

nombre(G,N) $\longrightarrow$ [P] , { lex (P, nombre ,G,N) } .

verbo(N) $\longrightarrow$ [P] , { lex (P, verbo ,N) \} .

Bibliografía

- P. Blackburn, J. Bos y K. Striegnitz *Learn Prolog Now!*
[<http://www.coli.uni-sb.de/~kris/learn-prolog-now>]
 - ▶ Cap. 7 “Definite Clause Grammars”
 - ▶ Cap. 8 “More Definite Clause Grammars”
- I. Bratko *Prolog Programming for Artificial Intelligence (Third ed.)* (Prentice–Hall, 2001)
 - ▶ Cap 21: “Language Processing with Grammar Rules”
- P. Flach *Simply Logical (Intelligent Reasoning by Example)* (John Wiley, 1994)
 - ▶ Cap. 7: “Reasoning with natural language”
- U. Nilsson y J. Maluszynski *Logic, Programming and Prolog (2nd ed.)*
[<http://www.ida.liu.se/~ulfni/lpp>]
 - ▶ Cap. 10 “Logic and grammars”