

Tema 9: Procesamiento de lenguaje natural

Grupo de Programación Declarativa

{José A. Alonso, Andrés Cordon y Miguel A. Gutiérrez}

Dpto. de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE SEVILLA

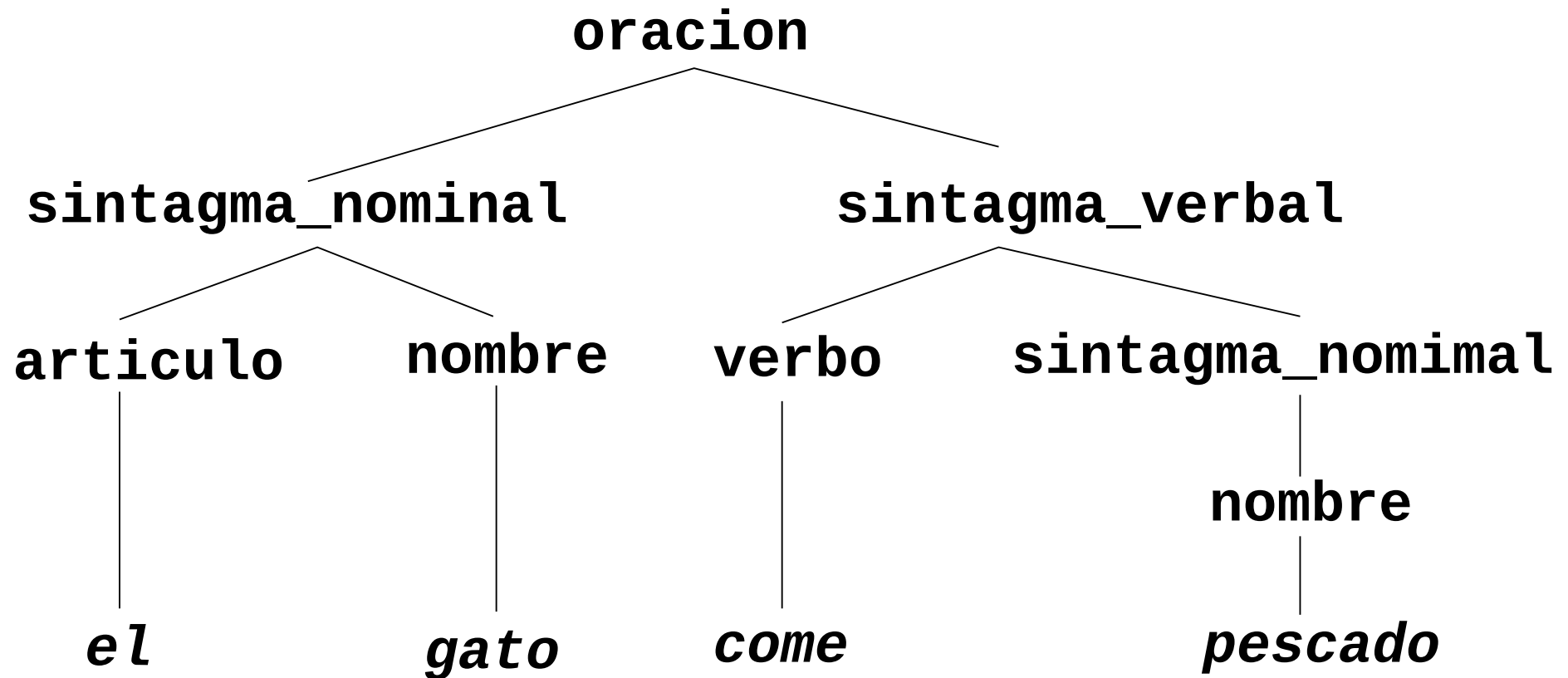
Gramáticas libres de contexto: Ejemplo

- Ejemplos de frases
 - El gato come pescado
 - El perro come carne
- Ejemplo de gramática libre de contexto (GLC)

oración	-->	sintagma_nominal, sintagma_verbal
sintagma_nominal	-->	nombre
sintagma_nominal	-->	artículo, nombre
sintagma_verbal	-->	verbo, sintagma_nominal
artículo	-->	[el]
nombre	-->	[gato]
nombre	-->	[perro]
nombre	-->	[pescado]
nombre	-->	[carne]
verbo	-->	[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
 - $xAy \Rightarrow xwy$ mediante $A \Rightarrow w$
 - $x \xRightarrow{*} y$ si existen $x_1, x_2, \dots, x_n$ tales que $x = x_1 \Rightarrow x_2 \cdots \Rightarrow x_{n-1} \Rightarrow x_n = y$
- Lenguaje definido por una gramática: $L(G) = \{x \in T^* : S \xRightarrow{*} x\}$
- Gramáticas libres de contextos (GLC): $A \Rightarrow w$ 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(0)                    :- sintagma_nominal(SN), sintagma_verbal(SV), append(SN,SV,0).  
sintagma_nominal(SN) :- nombre(SN).  
sintagma_nominal(SN) :- artículo(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(0).  
0 = [gato, come, gato] ;  
0 = [gato, come, perro] ;  
0 = [gato, come, pescado]  
Yes  
?- findall(_0,oración(_0),_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]]
```

Reconocedor de GLC mediante listas de diferencia

- Reconocedor de GLC en Prolog 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).  
sintagma_nominal(A-B) :- artículo(A-C), 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(0-[]).  
0 = [gato, come, gato] ;  
0 = [gato, come, perro] ;  
0 = [gato, come, pescado]  
Yes  
?- findall(_0,oración(_0-[]),_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	-->	sintagma_nominal, sintagma_verbal.
sintagma_nominal	-->	nombre.
sintagma_nominal	-->	artículo, nombre.
sintagma_verbal	-->	verbo, sintagma_nominal.
artículo	-->	[el].
nombre	-->	[gato].
nombre	-->	[perro].
nombre	-->	[pescado].
nombre	-->	[carne].
verbo	-->	[come].

Gramáticas de cláusulas definidas: Usos

- Usos de la gramática

- Reconocimiento de oraciones

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

```
Yes
```

```
?- oración([el,come,pescado],[]).
```

```
No
```

- Generación de las oraciones

```
?- oración(0,[]).
```

```
0 = [gato, come, gato] ;
```

```
0 = [gato, come, perro] ;
```

```
0 = [gato, come, pescado]
```

```
Yes
```

```
?- findall(_0,oración(_0,[]),_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]]
```

- Determinación de elementos

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

X = el

Y = come ;

No

- La relación phrase

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

Yes

```
?- phrase(sintagma_nominal,L).
```

```
L = [gato] ;
```

```
L = [perro]
```

Yes

Gramáticas de cláusulas definidas: Compilación

- **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([el|A], A).
nombre([gato|A], A).
nombre([perro|A], A).
nombre([pescado|A], A).
nombre([carne|A], A).
verbo([come|A], A).
Yes
```

- **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
```

Reglas recursivas en GCD

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

- GCD

oración	-->	oración, conjunción, oración.
oración	-->	sintagma_nominal, sintagma_verbal.
sintagma_nominal	-->	nombre.
sintagma_nominal	-->	artículo, nombre.
sintagma_verbal	-->	verbo, sintagma_nominal.
artículo	-->	[el].
nombre	-->	[gato].
nombre	-->	[perro].
nombre	-->	[pescado].
nombre	-->	[carne].
verbo	-->	[come].
conjunción	-->	[y].
conjunción	-->	[o].

Reglas recursivas en GCD

- 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

oración	-->	sintagma_nominal, sintagma_verbal.
oración	-->	oración, conjunción, oración.
sintagma_nominal	-->	nombre.
sintagma_nominal	-->	artículo, nombre.
sintagma_verbal	-->	verbo, sintagma_nominal.
artículo	-->	[el].
nombre	-->	[gato].
nombre	-->	[perro].
nombre	-->	[pescado].
nombre	-->	[carne].
verbo	-->	[come].
conjunción	-->	[y].
conjunción	-->	[o].

- 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

- GCD

```
oración          --> oración_simple.
oración          --> oración_simple, conjunción, oración.
oración_simple   --> sintagma_nominal, sintagma_verbal.
sintagma_nominal --> nombre.
sintagma_nominal --> artículo, nombre.
sintagma_verbal  --> verbo, sintagma_nominal.
artículo         --> [el].
nombre           --> [gato].
nombre           --> [perro].
nombre           --> [pescado].
nombre           --> [carne].
verbo            --> [come].
conjunción       --> [y].
conjunción       --> [o].
```

- 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] ;

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

Yes

- GCD

s --> [] .

s --> i,s,d.

i --> [a] .

d --> [b] .

Árbol de análisis con GCD

- Árbol de análisis con GCD

- Sesión

```
?- oración(A,[el,gato,come,pescado],[[]]).  
A = o(sn(art(el),n(gato)),sv(v(come),sn(n(pescado))))  
Yes
```

- Definición

oración(o(SN,SV))	-->	sintagma_nominal(SN), sintagma_verbal(SV).
sintagma_nominal(sn(N))	-->	nombre(N).
sintagma_nominal(sn(Art,N))	-->	artículo(Art), nombre(N).
sintagma_verbal(sv(V,SN))	-->	verbo(V), sintagma_nominal(SN).
artículo(art(el))	-->	[el].
nombre(n(gato))	-->	[gato].
nombre(n(perro))	-->	[perro].
nombre(n(pescado))	-->	[pescado].
nombre(n(carne))	-->	[carne].
verbo(v(come))	-->	[come].

Árbol de análisis con GCD

- Compilación

```
?- listing([oración,sintagma_nominal,sintagma_verbal,artículo,nombre,verbo]).  
  
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).  
  
sintagma_verbal(sv(A, B), C, D) :- verbo(A, C, E), sintagma_nominal(B, E, D).  
  
artículo(art(el), [el|A], A).  
  
nombre(n(gato), [gato|A], A).  
nombre(n(perro), [perro|A], A).  
nombre(n(pescado), [pescado|A], A).  
nombre(n(carne), [carne|A], A).  
  
verbo(v(come), [come|A], A).
```

Yes

Concordancia de género en GCD

- Concordancia de género en GCD

- Sesión

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

```
Yes
```

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

```
No
```

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

```
Yes
```

- Definición

```
oración                --> sintagma_nominal, sintagma_verbal.
```

```
sintagma_nominal       --> nombre(_).
```

```
sintagma_nominal       --> artículo(G), nombre(G).
```

```
sintagma_verbal        --> verbo, sintagma_nominal.
```

```
artículo(masculino)    --> [el].
```

```
artículo(femenino)     --> [la].
```

```
nombre(masculino)      --> [gato].
```

```
nombre(femenino)       --> [gata].
```

```
nombre(masculino)      --> [pescado].
```

```
verbo                  --> [come].
```

Concordancia de número en GCD

- Concordancia de número en GCD

- Sesión

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

```
Yes
```

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

```
No
```

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

```
Yes
```

- Definición

```
oración                -->  sintagma_nominal(N), sintagma_verbal(N).
```

```
sintagma_nominal(N)    -->  nombre(N).
```

```
sintagma_nominal(N)    -->  artículo(N), nombre(N).
```

```
sintagma_verbal(N)     -->  verbo(N), sintagma_nominal(_).
```

```
artículo(singular)     -->  [el].      artículo(plural)      -->  [los].
```

```
nombre(singular)       -->  [gato].     nombre(plural)        -->  [gatos].
```

```
nombre(singular)       -->  [perro].    nombre(plural)        -->  [perros].
```

```
nombre(singular)       -->  [pescado].
```

```
nombre(singular)       -->  [carne].
```

```
verbo(singular)        -->  [come].     verbo(plural)         -->  [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
```

- GCD

```
s                --> bloque_a(N), bloque_b(N), bloque_c(N).  
bloque_a(0)      --> [].  
bloque_a(suc(N)) --> [a], bloque_a(N).  
bloque_b(0)      --> [].  
bloque_b(suc(N)) --> [b], bloque_b(N).  
bloque_c(0)      --> [].  
bloque_c(suc(N)) --> [c], 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,c,c,c,c,c,c] ; ...
```

```
Yes
```

- GCD

```
s                --> bloque_a(N), bloque_b(N), bloque_c(N), {par(N)}.
```

```
bloque_a(0)      --> [].
```

```
bloque_a(s(N))   --> [a], bloque_a(N).
```

```
bloque_b(0)      --> [].
```

```
bloque_b(s(N))   --> [b], bloque_b(N).
```

```
bloque_c(0)      --> [].
```

```
bloque_c(s(N))   --> [c], bloque_c(N).
```

```
par(0).
```

```
par(s(s(N))) :- par(N).
```

Separación de reglas y lexicon

- Gramática

- Lexicón

```
lex(el,artículo).  
lex(gato,nombre).  
lex(perro,nombre).  
lex(pescado,nombre).  
lex(carne,nombre).  
lex(come,verbo).
```

- Regla

```
oración      --> sintagma_nominal, sintagma_verbal.  
sintagma_nominal --> nombre.  
sintagma_nominal --> artículo, nombre.  
sintagma_verbal  --> verbo, sintagma_nominal.  
artículo        --> [Palabra], {lex(Palabra,artículo)}.  
nombre          --> [Palabra], {lex(Palabra,nombre)}.  
verbo           --> [Palabra], {lex(Palabra,verbo)}.
```

- Sesión

```
?- oración([el,gato,come,pescado],[]).  
Yes  
?- oración([el,come,pescado],[]).  
No
```


Separación de reglas y lexicon con concordancia

- Sesión

```
?- 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).
```

- Reglas

```
oración          -->  sintagma_nominal(N), sintagma_verbal(N).
sintagma_nominal(N) -->  nombre(N).
sintagma_nominal(N) -->  artículo(N), nombre(N).
sintagma_verbal(N) -->  verbo(N), sintagma_nominal(_).
artículo(N)       -->  [Palabra], {lex(Palabra,artículo,N)}.
nombre(N)          -->  [Palabra], {lex(Palabra,nombre,N)}.
verbo(N)           -->  [Palabra], {lex(Palabra,verbo,N)}.
```

Lexicón con género y número

- 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

```
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).                    lex(leen,verbo,plural).
```

Lexicón con género y número

- Reglas

```
oración          --> sintagma_nominal(N), verbo(N), complemento.
complemento      --> [].
complemento      --> sintagma_nominal(_).
sintagma_nominal(N) --> nombre(_,N).
sintagma_nominal(N) --> determinante(G,N), nombre(G,N).
determinante(G,N) --> [P], {lex(P,determinante,G,N)}.
nombre(G,N)      --> [P], {lex(P,nombre,G,N)}.
verbo(N)         --> [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”

Bibliografía

- U. Nilsson y J. Maluszynski *Logic, Programming and Prolog (2nd ed.)* (Autores, 2000) [<http://www.ida.liu.se/~ulfni/lpp>]
 - Cap. 10 “Logic and grammars”
- L. Sterling y E. Shapiro *The Art of Prolog (2nd edition)* (The MIT Press, 1994)
 - Cap. 19: “Logic grammars”