

# Tema 5: Procesamiento de lenguaje natural

José A. Alonso Jiménez  
Miguel A. Gutiérrez Naranjo  
Francisco J. Martín Mateos

Dpto. de Ciencias de la Computación e Inteligencia Artificial  
UNIVERSIDAD DE SEVILLA

# Contenido

- Procesamiento de lenguaje natural
  - Gramáticas libres de contexto en Prolog:
    - \* Concepto de GLC
    - \* GLC en Prolog mediante listas
    - \* GLC en Prolog mediante listas de diferencia
  - Gramáticas de cláusulas definidas:
    - \* Metaintérprete para GCD
    - \* GCD de Prolog
    - \* GCD con árbol de análisis
    - \* GCD con concordancia de género y número
    - \* GCD con semántica
  - Razonamiento con lenguaje natural
  - Bibliografía fundamental:
    - P. Flach “Simply Logical (Intelligent Reasoning by Example)” (John Wiley, 1994)
    - Cap. 7: “Reasoning with natural lenguaje”

# Gramáticas libres de contexto

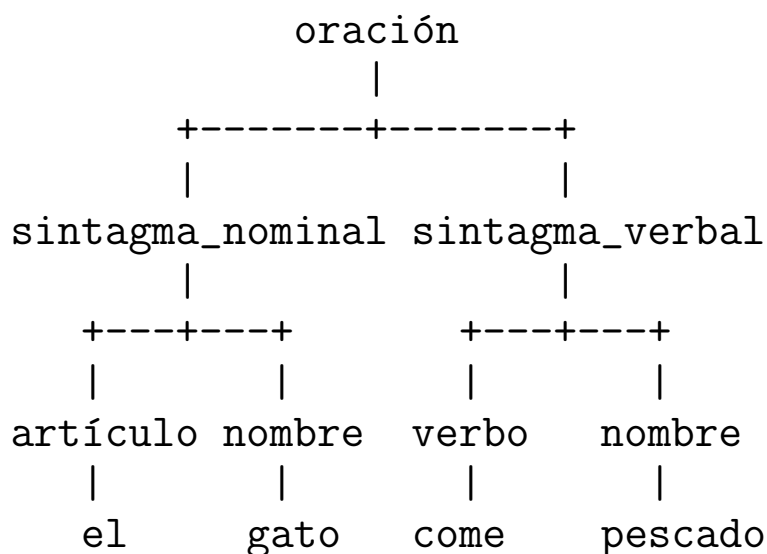
- Ejemplos de frases

- El gato come pescado
- El perro come carne

- Ejemplo de gramática

```
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]
```

- Arbol de análisis



# Gramáticas libres de contexto

- 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
  - $N \cap T = \emptyset$
- Gramáticas libres de contextos  
 $A \Longrightarrow w$  con  $A \in N$  y  $w \in V^*$
- Derivaciones
  - $xAy \Longrightarrow xwy$  mediante  $A \Longrightarrow w$
  - $x \xRightarrow{*} y$  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  
 $L(G) = \{x \in T^* : S \xRightarrow{*} x\}$

# Gramáticas libres de contexto en Prolog

- Representación de oraciones en Prolog

[el, gato, come, pescado]

[el, perro, come, carne]

- Gramática en Prolog con listas

- Sesión

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

Yes

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

No

- Definición

```
oracion(0) :-          sintagma_nominal(SN),
                      sintagma_verbal(SV),
                      append(SN,SV,0).
```

```
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).
```

```
articulo([el]).
```

```
nombre([gato]).
```

```
nombre([perro]).
```

```
nombre([pescado]).
```

```
nombre([carne]).
```

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

# Gramáticas libres de contexto en Prolog

- Gramática en Prolog con listas de diferencia

- Sesión

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

Yes

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

No

- Definición

```
oracion(L-L0) :-  
    sintagma_nominal(L-L1),  
    sintagma_verbal(L1-L0).
```

```
sintagma_nominal(L-L0) :-  
    nombre(L-L0).
```

```
sintagma_nominal(L-L0) :-  
    articulo(L-L1),  
    nombre(L1-L0).
```

```
sintagma_verbal(L-L0) :-  
    verbo(L-L1),  
    sintagma_nominal(L1-L0).
```

```
articulo([el|L]-L).  
nombre([gato|L]-L).  
nombre([perro|L]-L).  
nombre([pescado|L]-L).  
nombre([carne|L]-L).  
verbo([come|L]-L).
```

# Gramáticas de cláusulas definidas

- Metaintérprete para GCD

- Gramática

```
:- op(1200,xfx,--->).
```

```
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].
```

- Sesión

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

- Metaintérprete de GCD

```
deriva([],E-E).
deriva([X],[X|E]-E).
deriva((X,Y),E0-E2) :-
    deriva(X,E0-E1), deriva(Y,E1-E2).
deriva(X,E0-E1) :-
    (X ---> Y),
    deriva(Y,E0-E1).
```

# Gramáticas de cláusulas definidas

## • 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].
```

### • 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).
```



# Gramáticas de cláusulas definidas

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

## • Consulta

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

Yes

?- oración([el,come,pescado], []).

No

?- oración([el,gato,X,pescado], []).

X = come ;

No

?- oración([X,gato,Y,pescado], []).

X = el

Y = come ;

No

?- sintagma\_nominal(L, []).

L = [gato] ;

L = [perro]

Yes

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

Yes

?- phrase(sintagma\_nominal,L).

L = [gato] ;

L = [perro]

Yes

# Gramáticas de cláusulas definidas

## • Arbol de análisis con GCD

### • Sesión

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

### • 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).
```

# Gramáticas de cláusulas definidas

```
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

# Gramáticas de cláusulas definidas

- **Concordancia de género**

- **Sesión**

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

Yes

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

No

?- phrase(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].

# Gramáticas de cláusulas definidas

## ● Concordancia en número

### ● Sesión

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

Yes

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

No

?- phrase(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].

# Gramáticas de cláusulas definidas

- GCD con llamadas a Prolog

- $L = \{a^{2n}b^{2n}c^{2n} : n \in \mathbb{N}\}$

- Ejemplos

```
?- palabra([a,a,b,b,c,c], []).
```

```
Yes
```

```
?- palabra([a,b,c], []).
```

```
No
```

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

```
L = [] ;
```

```
L = [a,a,b,b,c,c] ;
```

```
L = [a,a,a,a,b,b,b,b,c,c,c,c] ;
```

```
L = [a,a,a,a,a,a,b,b,b,b,b,b,c,c,c,c,c,c]
```

```
Yes
```

- Gramática

```
palabra --> a(N), b(N), c(N), {par(N)}.
```

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

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

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

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

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

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

```
par(0).
```

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

- Compilación

```
?- listing(palabra).
```

```
palabra(A, B) :-
```

```
    a(C, A, D), b(C, D, E), c(C, E, F),
```

```
    par(C), B=F.
```

# Gramáticas de cláusulas definidas

## • Sesión

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

Yes

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

No

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

Yes

## • Definición

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

sintagma_nominal(N)    -->  nombre(N).
sintagma_nominal(N)    -->  articulo(N),
                           nombre(N).

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

articulo(singular)     -->  [el].
articulo(plural)       -->  [los].
verbo(singular)        -->  [come].
verbo(plural)          -->  [comen].
nombre(singular)       -->  [Palabra],
                           {es_nombre(Palabra, _)}.
nombre(plural)         -->  [Palabra],
                           {es_nombre(_, Palabra)}.

es_nombre(gato,        gatos).
es_nombre(perro,       perros).
es_nombre(pescado,     pescados).
es_nombre(carne,       carnes).
```

# Gramáticas de cláusulas definidas

## ● Concordancia en género y número

### ● Sesión

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

Yes

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

No

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

Yes

?- phrase(oración, [los, profesores, leen]).

Yes

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

Yes

### ● Definición

es\_nombre(profesor, masculino, singular).

es\_nombre(profesores, masculino, plural).

es\_nombre(profesora, femenino, singular).

es\_nombre(profesoras, femenino, plural).

es\_nombre(libro, masculino, singular).

es\_nombre(libros, masculino, plural).

es\_determinante(el, masculino, singular).

es\_determinante(los, masculino, plural).

es\_determinante(la, femenino, singular).

es\_determinante(las, femenino, plural).

es\_determinante(un, masculino, singular).

es\_determinante(una, femenino, singular).

es\_determinante(unos, masculino, plural).

es\_determinante(unas, femenino, plural).

es\_verbo(lee, singular).

es\_verbo(leen, plural).



# Gramáticas de cláusulas definidas

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).
verbo(N)	--> [P], {es_verbo(P,N)}.
nombre(G,N)	--> [P], {es_nombre(P,G,N)}.
determinante(G,N)	--> [P], {es_determinante(P,G,N)}.

# Razonamiento con lenguaje natural

- Gramática de asertos y preguntas

- Ejemplos

```
?- phrase(oración(O),L).  
O = europeo(juan) :- true  
L = [juan, es, europeo] ;  
O = andaluz(juan) :- true  
L = [juan, es, andaluz] ;  
O = europeo(_G273) :- europeo(_G273)  
L = [todo, europeo, es, europeo] ;  
O = andaluz(_G273) :- europeo(_G273)  
L = [todo, europeo, es, andaluz] ;  
O = europeo(_G273) :- andaluz(_G273)  
L = [todo, andaluz, es, europeo] ;  
O = andaluz(_G273) :- andaluz(_G273)  
L = [todo, andaluz, es, andaluz] ;  
No
```

```
?- phrase(pregunta(P),L).  
P = europeo(juan)  
L = [¿, es, juan, europeo, ?] ;  
P = andaluz(juan)  
L = [¿, es, juan, andaluz, ?] ;  
P = europeo(_G297)  
L = [¿, quién, es, europeo, ?] ;  
P = andaluz(_G297)  
L = [¿, quién, es, andaluz, ?] ;  
No
```

# Razonamiento con lenguaje natural

- Definición

```
:- op(600,xfy, '=>').
```

```
oración((L:-true))    --> nombre_propio(X),
                        sintagma_verbal(X=>L).
oración(C)            --> determinante(A1,A2,C),
                        adjetivo(A1),
                        sintagma_verbal(A2).
sintagma_verbal(A)    --> verbo, adjetivo(A).

pregunta(P)           --> [¿,es],
                        nombre_propio(X),
                        adjetivo(X=>P),
                        [?].
pregunta(P)           --> [¿,quién,es],
                        adjetivo(_X=>P),
                        [?].

nombre_propio(juan)   --> [juan].
determinante(X=>Cu,X=>Ca,(Ca:-Cu)) --> [todo].
verbo                 --> [es].
adjetivo(X=>europeo(X)) --> [europeo].
adjetivo(X=>andaluz(X)) --> [andaluz].
```

# Razonamiento con lenguaje natural

- Sistema de consulta y razonamiento

- Ejemplo

```
?- consulta([]).
? [juan,es,andaluz].
? [j, quién, es, andaluz, ?].
! [juan, es, andaluz]
? [j, es, juan, europeo, ?].
! No
? [todo, andaluz, es, europeo].
? [j, es, juan, europeo, ?].
! [juan, es, europeo]
? [j, quién, es, europeo, ?].
! [juan, es, europeo]
? muestra_reglas.
! [todo, andaluz, es, europeo]
! [juan, es, andaluz]
? fin.
```

Yes

- Definición

```
consulta(Base_de_reglas) :-
    pregunta_y_lee(Entrada),
    procesa_entrada(Entrada,Base_de_reglas).
```

```
pregunta_y_lee(Entrada) :-
    write('? '),
    read(Entrada).
```

# Razonamiento con lenguaje natural

```
procesa_entrada(fin,_Base_de_reglas) :- !.
procesa_entrada(muestra_reglas,Base_de_reglas) :- !,
    muestra_reglas(Base_de_reglas),
    consulta(Base_de_reglas).
procesa_entrada(Oración,Base_de_reglas) :-
    phrase(oración(Regla),Oración), !,
    consulta([Regla|Base_de_reglas]).
procesa_entrada(Pregunta,Base_de_reglas) :-
    phrase(pregunta(P),Pregunta),
    prueba(P,Base_de_reglas), !,
    transforma(P,Clausula),
    phrase(oración(Clausula),Respuesta),
    muestra_respuesta(Respuesta),
    consulta(Base_de_reglas).
procesa_entrada(_Pregunta,Base_de_reglas) :-
    muestra_respuesta('No'),
    consulta(Base_de_reglas).

muestra_reglas([]).
muestra_reglas([Regla|Reglas]) :-
    phrase(oración(Regla),Oración),
    muestra_respuesta(Oración),
    muestra_reglas(Reglas).

muestra_respuesta(Respuesta) :-
    write('! '),
    write(Respuesta),
    nl.
```

# Razonamiento con lenguaje natural

```
prueba(true, _Base_de_reglas) :- !.
prueba((A,B), Base_de_reglas) :- !,
    prueba(A, Base_de_reglas),
    prueba(B, Base_de_reglas).
prueba(A, Base_de_reglas) :-
    busca_clausula((A:-B), Base_de_reglas),
    prueba(B, Base_de_reglas).

busca_clausula(Clausula, [Regla|_Reglas]) :-
    copy_term(Regla, Clausula).
busca_clausula(Clausula, [_Regla|Reglas]) :-
    busca_clausula(Clausula, Reglas).

transforma((A,B), [(A:-true)|Resto]) :- !,
    transforma(B, Resto).
transforma(A, (A:-true)).
```

## Bibliografía

- Bratko, I. *Prolog Programming for Artificial Intelligence (Third ed.)* (Prentice–Hall, 2001)
  - Cap 21: “Language Processing with Grammar Rules”
- Cortés, U. et als. *Inteligencia artificial* (Ediciones UPC, 1993)
  - Cap. 10: “Tratamiento del lenguaje natural”
- Flach, P. *Simply Logical (Intelligent Reasoning by Example)* (John Wiley, 1994)
  - Cap. 7: “Reasoning with natural language”
- Pereira, F.C. y Shieber, S.M. *Prolog and natural-languages analysis* (CSLI, 1987)
- Rich, E. y Knight, K. *Inteligencia artificial (segunda edición)* (McGraw–Hill Interamericana, 1994).
  - Cap. 15: “Procesamiento del lenguaje natural”
- Russell, S. y Norvig, P. *Inteligencia artificial (un enfoque moderno)* (Prentice Hall, 1996)
  - Cap. 22: “Agentes que se comunican”
  - Cap. 23: “Procesamiento práctico del lenguaje natural”