Tema 3: Procesamiento de lenguaje natural

José A. Alonso Jiménez
Francisco Jesús Martín Mateos
José Luis Ruiz Reina

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Procesamiento del lenguaje natural

- **Lenguaje natural: es el lenguaje utilizado por los seres humanos para comunicarse**
  - **En contraposición a los lenguajes formales (programación, lógica...)**
- **Procesamiento: comprensión y generación**
  - **Comprensión: a partir de una frase escrita o hablada, obtener una representación formal que permita efectuar las acciones adecuadas a la información recibida**
  - **Generación: transformar una representación formal de algo que se quiere comunicar, a una expresión en algún lenguaje natural, escrita o hablada**
- **En este tema veremos algunos aspectos básicos acerca de la *comprensión* de fragmentos simples del castellano**
  - **Para ello, usaremos herramientas que se emplean en el estudio de lenguajes formales**

Comprensión del lenguaje natural

- **Fases en el proceso de comprensión:**
  - **Percepción: reconocimiento del habla y/o escritura**
  - **Análisis: sintáctico (obtener la estructura de una frase a partir de la secuencia de palabras) y semántico (obtener un significado a partir de la estructura sintáctica)**
  - **Eliminación de ambigüedades: escoger uno de los posibles significados**
  - **Incorporación a la base de conocimiento**
- **En lo que sigue, nos centraremos en el análisis sintáctico y semántico de frases escritas**

Gramáticas formales

- **Gramáticas formales: $G = (N, T, P, S)$**
  - **N: símbolos no terminales (categorías sintácticas)**
  - **T: símbolos terminales (palabras del idioma), con $N \cap T = \emptyset$**
  - **P: conjunto de *reglas de producción* $l \rightarrow r$, donde $l, r \in (N \cup T)^*$**
  - **$S \in N$: símbolo inicial**
- **En una gramática *independiente del contexto (GIC)*, las reglas son de la forma $A \rightarrow r$, con $A \in N$ y $r \in (N \cup T)^*$**
  - **Derivación: $xAy \Longrightarrow xwy$ mediante la regla $A \rightarrow w$**
- **Lenguaje definido por una gramática: $L(G) = \{x \in T^* : S \overset{*}{\Longrightarrow} x\}$**
- **El problema del reconocimiento de frases y del análisis sintáctico:**
  - **Dado $x \in T^*$ y una gramática G, *decidir* si $x \in L(G)$, encontrando una derivación (*arbol de análisis sintáctico*) que conecta S con x**

Notación Prolog para gramáticas formales

- **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].
```

- **Notación Prolog para las gramáticas:**
  - **uso de `-->`**
    y punto al final de cada regla
  - **símbolos terminales entre corchetes**

Notación Prolog para gramáticas formales

- **¿ Por qué escribir las gramáticas en notación Prolog ?**
  - **porque pueden ser cargadas directamente en Prolog**
  - **disponiendo así de un analizador sintáctico ejecutable de manera inmediata**
  - **e incluso de un *generador* de frases del lenguaje**
- **Ejemplo:**

```
?- [g1].
% g1 compiled 0.01 sec, 2,292 bytes
Yes
?- phrase(oración,[el,gato,come,carne]).
Yes
?- phrase(oración,[gato,come,pescado,carne]).
No
?- phrase(sintagma_verbal,[come,pescado]).
Yes
?- phrase(oración,X).
X = [gato, come, gato] ;
X = [gato, come, perro] ;
X = [gato, come, pescado] ;
...
```

El programa Prolog generado por una gramática

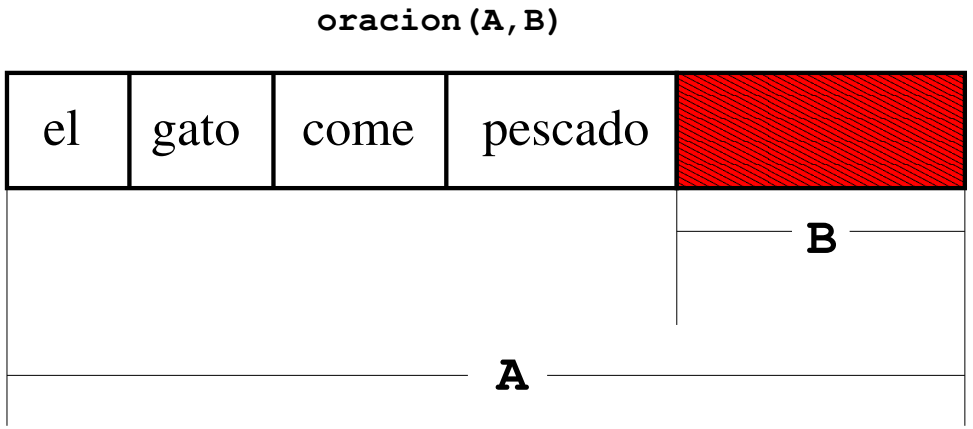
- **La gramática anterior genera, de manera automática, el programa:**

```
?- 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).
```

- **Transformaciones:**
  - **cada regla de la gramática genera una cláusula**
  - **cada símbolo no terminal se corresponde con un predicado**
  - **cada uno de estos predicados tiene dos argumentos añadidos**

El programa Prolog generado por una gramática

- El papel de los dos argumentos añadidos: diferencia de listas
  - Por ejemplo, `oración(A,B)` es un predicado que se cumple cuando `A` es una lista de símbolos terminales que contiene una *parte inicial* que puede ser reconocida como oración y el resto de símbolos queda en `B`



El programa Prolog generado por una gramática

- **Ejemplo:**

```
?- oración([el,gato,come,pescado,1,qwerty,3216],[1,qwerty,3216]).
Yes
?- oración([el,gato,come,pescado|X],X).
X = _G157
?- oración([el,gato,come,pescado],[]).
Yes
```

  - `phrase(<símbolo>,<lista>)` **es una abreviatura de** `<símbolo>(<lista>,[])`

Algoritmos de análisis sintáctico

- **Analizador sintáctico implementado por el programa anterior:**
  - **Esencialmente, usa el mecanismo de backtracking y de unificación incorporado en Prolog**
- **Existen algoritmos de análisis sintáctico más eficientes:**
  - **Análisis ascendente**
  - **Análisis descendente**
- **No es objetivo de este tema el estudio de dichos algoritmos**
  - **Asignatura "Procesadores de Lenguajes I" (cuarto curso)**

Gramáticas de cláusulas definidas

- **Las gramáticas que se pueden escribir en Prolog son más *expresivas* que las GIC, ya que permiten incluir:**
  - **argumentos en los símbolos no terminales**
  - **llamadas directas a predicados Prolog**
- **Con esta ampliación, las GIC se denominan gramáticas de cláusulas definidas (GCD)**
- **La capacidad expresiva de las GCD es mayor que la de las GIC**
  - **Existen lenguajes descritos por una GCD que no es posible describir mediante una GIC**

Ejemplo de GCD

- **La siguiente GCD (que incluye argumentos y llamadas a Prolog) define el lenguaje $L = \{a^{2n}b^{2n}c^{2n} : n \in \mathbb{N}\}$ (no expresable con una GIC):**

```
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).
```

- **Ejemplo de sesión:**

```
?- palabra([a,a,b,b,c,c],[]).
Yes
?- palabra([a,b,c],[]).
No
?- phrase(palabra,L).
L = [] ;
L = [a,a,b,b,c,c] ;
...
```

Ejemplo de GCD

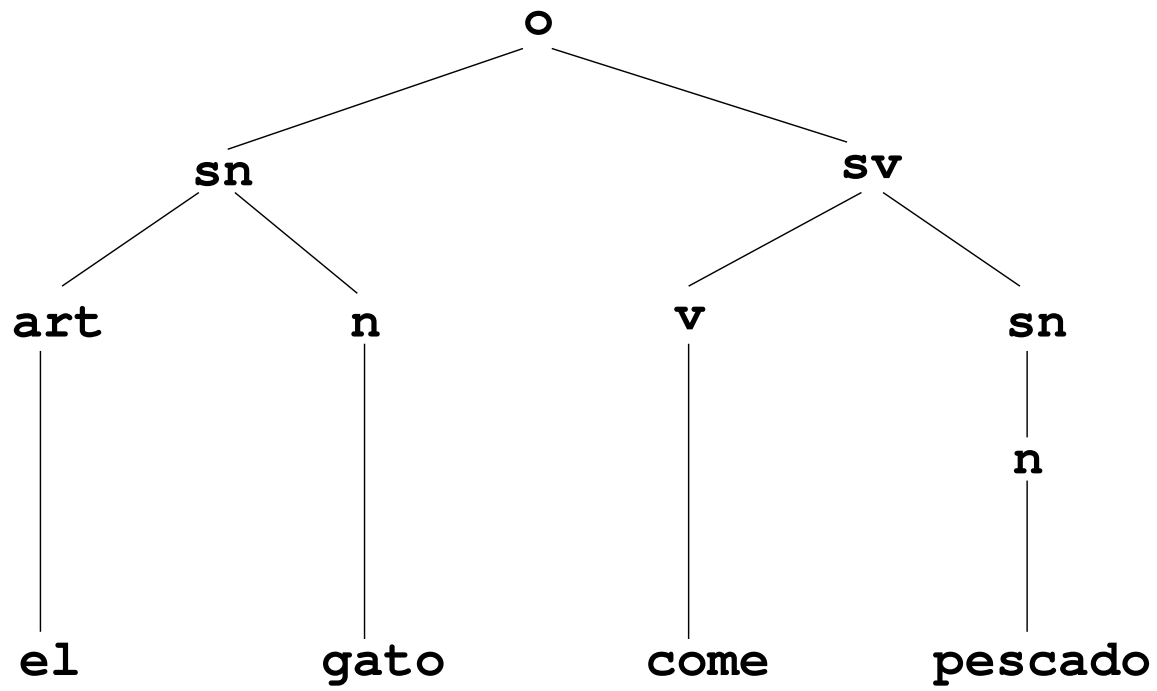
- **Programa Prolog generado:**

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

palabra(A, B) :- a(C, A, D),b(C, D, E),c(C, E, B),par(C).
a(0, A, A).
a(s(A), B, C) :- 'C'(B, a, D),a(A, D, C).
b(0, A, A).
b(s(A), B, C) :- 'C'(B, b, D),b(A, D, C).
c(0, A, A).
c(s(A), B, C) :-'C'(B, c, D),c(A, D, C).
'C'([A|B], A, B).
```

- **Observaciones:**
  - **El predicado correspondiente a cada símbolo no terminal se aumenta con los dos argumentos anteriormente mencionados**
  - **Las llamadas a predicados Prolog se incluyen directamente**

Árboles sintácticos

- **Un árbol sintáctico refleja cómo los componentes de una frase se estructuran en diferentes categorías sintácticas:**



GCD para construcción de árboles sintácticos

- **El árbol sintáctico como argumento adicional de una GCD**

```
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].
```

- **Sesión**

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

GCD para concordancia de género y número

- **Una GIC no puede describir las concordancias de género y número**
  - **Sin embargo, podemos usar argumentos para controlar estas concordancias**
  - **Por tanto, podemos definir la concordancia de género y número con una GCD**
- **Ejemplo:**

```
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)}.
```

GCD para concordancia de género y número

- **Ejemplo (continuación):**

```
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).
```

GCD para concordancia de género y número

- **Ejemplo (sesión):**

```
?- phrase(oración,[la,profesora,lee,un,libro]).
Yes
?- phrase(oración,[la,profesor,lee,un,libro]).
No
?- phrase(oración,[las,profesores,lee,los,libro]).
No
?- phrase(oración,[los,profesores,leen,un,libro]).
Yes
?- phrase(oración,[los,profesores,leen]).
Yes
?- phrase(oración,[los,profesores,leen,libros]).
Yes
```

Análisis semántico

- ¿Cómo representar el *significado* de una frase?
  - La respuesta a esta pregunta depende de lo que se pretenda hacer con el significado
  - En general, se trata de expresar el significado en algun lenguaje formal
  - Esto permite que una máquina pueda realizar las acciones adecuadas al mensaje emitido (almacenar información, responder preguntas razonadamente,...)
  - En nuestro caso, usaremos la lógica de primer orden como lenguaje de representación
  - Podríamos usar cualquier otro formalismo de representación

Análisis semántico

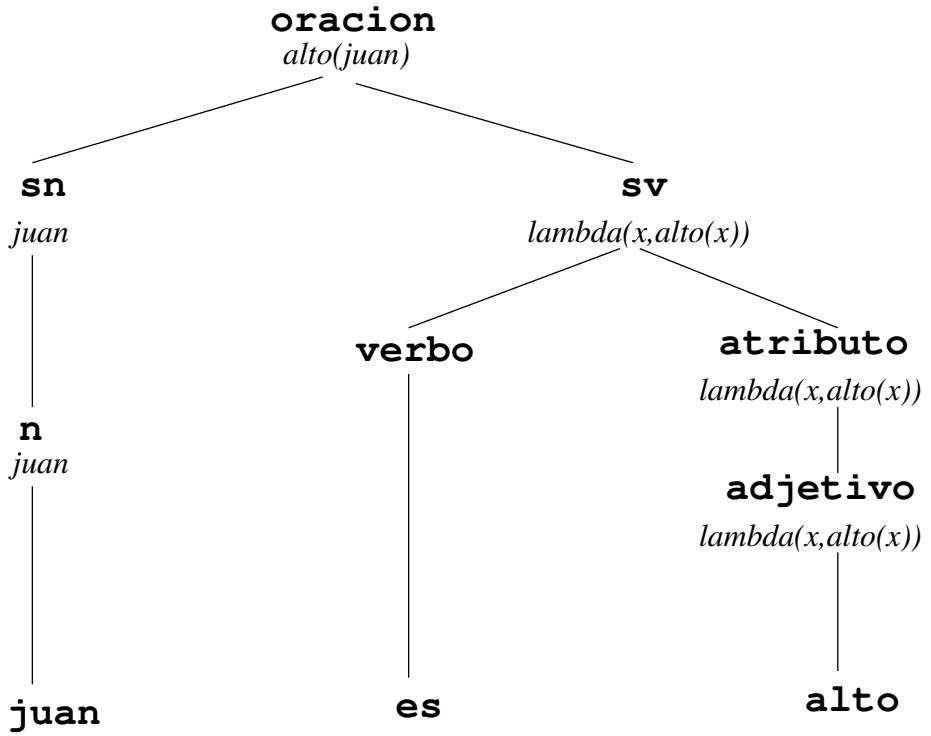
- **Ejemplos de significados asignados a frases:**
  - **Juan es alto:** $alto(juan)$
  - **Pedro bebe agua:** $bebe(pedro, agua)$
  - **Todo hombre tiene alma:** $\forall x[hombre(x) \rightarrow tiene(x, alma)]$
  - **Algun hombre tiene dinero:** $\exists x[hombre(x) \wedge tiene(x, dinero)]$
  - **Todo hombre que no come pan no tiene dinero:**
    $\forall x[(hombre(x) \wedge \neg come(x, pan)) \rightarrow \neg tiene(x, dinero)]$
  - **Si algún hombre tiene dinero y todo hombre que no come pan no tiene dinero entonces algun hombre come pan:**
    $(\exists x[hombre(x) \wedge tiene(x, dinero)] \wedge$
    $\forall x[(hombre(x) \wedge \neg come(x, pan)) \rightarrow \neg tiene(x, dinero)]) \rightarrow$
    $\exists x[hombre(x) \wedge come(x, pan)]$

Un caso simple de construcción de significado

- **¿Cuál es el significado de la frase "Juan es alto"?**
  - **Significado de "Juan": el término (constante)** `juan`
  - **Significado de "es": es sólo un nexo de unión del sujeto con el adjetivo que lo califica (no aporta significado)**
  - **Significado de "alto": predicado unario** `alto` **que expresa una propiedad sobre *alguien*; puede verse como una *función* tal que dado un *sujeto*, devuelve la afirmación de que dicho sujeto es alto; dicha función se representa usualmente por** `lambda(x,alto(x))`
  - **El significado de la frase completa se obtiene *aplicando* el significado del sintagma verbal al significado del sintagma nominal:**
    `lambda(x,alto(x))(juan) = alto(juan)`

Semántica composicional

- **Hipótesis composicional: el significado de una categoría sintáctica se obtiene a partir del significado de las subcategorías que lo componen**
  - **esta hipótesis no siempre es cierta, pero simplifica el análisis semántico**
  - **pasando parte del trabajo a la fase de eliminación de ambigüedades**



GCD para extracción de significado

- **GCD combinando el análisis sintáctico con el análisis semántico**
  - **para frases como las del ejemplo anterior**
  - **hipótesis composicional**

```
oración(SO) --> sintagma_nominal(SSN),sintagma_verbal(SSV),
                {componer(SSN,SSV,SO)}.
sintagma_nominal(SNP) --> nombre_propio(SNP).
sintagma_verbal(SA) --> verbo_cop,atributo(SA).
atributo(SA) --> adjetivo(SA).
verbo_cop --> [es].
nombre_propio(juan) --> [juan].
nombre_propio(pedro) --> [pedro].
adjetivo(lambda(X,alto(X))) --> [alto].
adjetivo(lambda(X,bajo(X))) --> [bajo].

componer(X,lambda(X,P),P).
```

  - **Sesión:**

```
?- phrase(oración(S),[juan,es,alto]).
S = alto(juan)
```

GCD para extracción de significado

- **Usualmente, se prescide del predicado `componer`**
  - **Para ello, los argumentos de la función `lambda` pasan a ser argumentos de los símbolos no terminales**
- **Gramática equivalente a la anterior:**

```
oración(SSV) --> sintagma_nominal(SSN), sintagma_verbal(SSN,SSV).
sintagma_nominal(SNP) --> nombre_propio(SNP).
sintagma_verbal(X,SA) --> verbo_cop,atributo(X,SA).
atributo(X,SA) --> adjetivo(X,SA).
verbo_cop --> [es].
nombre_propio(juan) --> [juan].
nombre_propio(pedro) --> [pedro].
adjetivo(X,alto(X)) --> [alto].
adjetivo(X,bajo(X)) --> [bajo].
```

  - **Sesión:**

```
?- phrase(oración(S),[juan,es,alto]).
S = alto(juan)
```

GCD para frases con verbos transitivos

- **Significado de un verbo transitivo: predicado que relaciona el sujeto con el objeto directo**
  - **Por ejemplo, el significado del verbo "come" es la *función*** `lambda(x,lambda(y,come(x,y)))`
- **Gramática:**

```
oración(SSV) --> sujeto(SS), sintagma_verbal(SS,SSV).
sujeto(SNP) --> nombre_propio(SNP).
sintagma_nominal(SN) --> nombre(SN).
sintagma_verbal(X,SV) --> verbo_trans(X,SN,SV),sintagma_nominal(SN).
verbo_trans(X,Y,come(X,Y)) --> [come].
verbo_trans(X,Y,bebe(X,Y)) --> [bebe].
nombre_propio(juan) --> [juan].
nombre_propio(pedro) --> [pedro].
nombre(pan) --> [pan].
nombre(agua) --> [agua].
```

  - **Sesión:**

```
?- phrase(oración(S),[pedro,come,pan]).
S = come(pedro, pan)
```

Frases con determinantes `todo` y `algún`

- **Definimos a continuación una GCD para obtener el significado de frases que empiezan por los determinantes `todo` y `algún`**
  - **En la lógica de primer orden, estos determinates se corresponden con los cuantificadores universal y existencial, respectivamente**
- **Ejemplos:**
  - **"Todo andaluz come pescado":** $\forall x[andaluz(x) \rightarrow come(x, pescado)]$
  - **"Algún informático tiene dinero":** $\exists x[informatico(x) \wedge tiene(x, dinero)]$
- **Sesión:**

```
?- phrase(oración(S),[todo,andaluz,come,pescado]).
S = para_todo(_G231, andaluz(_G231) => come(_G231, pescado))

?- phrase(oración(S),[algún,informático,tiene,dinero]).
S = existe(_G231, informático(_G231) y tiene(_G231, dinero))

?- phrase(oración(S),[algún,informático,es,andaluz]).
S = existe(_G231, informático(_G231) y andaluz(_G231))
```

GCD para frases con determinantes `todo` y `algún` (I)

```
:-op(600,xfy,'=>').
:-op(900,xfy,y).

oración(S) --> sujeto_det(X,SSV,S), sintagma_verbal(X,SSV).

sujeto_det(X,SSV,S) --> determinante(X,Prop,SSV,S),nombre_propiedad(X,Prop).

determinante(X,Prop,SSV,existe(X, Prop y SSV)) --> [algún].
determinante(X,Prop,SSV,para_todo(X, Prop => SSV)) --> [todo].

objeto_directo(SN) --> nombre(SN).

sintagma_verbal(X,SV) --> verbo_trans(X,SN,SV),objeto_directo(SN).
sintagma_verbal(X,SV) --> verbo_cop,nombre_propiedad(X,SV).
```

GCD para frases con determinantes `todo` y `algún` (II)

```
verbo_trans(X,Y,tiene(X,Y)) --> [tiene].
verbo_trans(X,Y,come(X,Y)) --> [come].

verbo_cop --> [es].

nombre(pan) --> [pan].
nombre(pescado) --> [pescado].
nombre(carne) --> [carne].
nombre(dinero) --> [dinero].
nombre(coche) --> [coche].

nombre_propiedad(X,hombre(X)) --> [hombre].
nombre_propiedad(X,carpintero(X)) --> [carpintero].
nombre_propiedad(X,informático(X)) --> [informático].
nombre_propiedad(X,andaluz(X)) --> [andaluz].
nombre_propiedad(X,francés(X)) --> [frances].
nombre_propiedad(X,europeo(X)) --> [europeo].
```

GCD para traducción inversa y generación de lenguaje

- **Una característica interesante de las GCD es que pueden servir, además, para generar lenguaje natural a partir de significados formales**
  - **Ejemplos:**

```
?- phrase(oración(existe(x, hombre(x) y tiene(x, dinero))),L).
L = [algún, hombre, tiene, dinero]

?- phrase(oración(S),L).

S = existe(_G354, hombre(_G354)y tiene(_G354, pan))
L = [algún, hombre, tiene, pan] ;

S = existe(_G354, hombre(_G354)y tiene(_G354, pescado))
L = [algún, hombre, tiene, pescado] ;

S = existe(_G354, hombre(_G354)y tiene(_G354, carne))
L = [algún, hombre, tiene, carne] ;

S = existe(_G354, hombre(_G354)y tiene(_G354, dinero))
L = [algún, hombre, tiene, dinero]
...
```

Interpretación pragmática

- **En la práctica, el análisis semántico descrito anteriormente puede no servir para extraer el significado de algunas frases**
  - **La *interpretación pragmática* es la fase del análisis semántico que tiene en cuenta la situación actual para construir el significado**
- **Ejemplos:**
  - **Adverbios: *ahora*, *ayer*, *mañana*,...**
  - **Pronombres: *yo*, *tú*, *él*,...**
- **La interpretación pragmática extrae el significado de estas palabras a partir de:**
  - **Descripciones de la situación actual (quién habla, cuándo habla, dónde está,...)**
  - **Frases pronunciadas anteriormente**

Ambigüedad en el lenguaje

- **Ambigüedad en el lenguaje**
  - **La misma frase puede tener diferentes significados**
  - **Ejemplos: *"Lo ví andando", "Pensé que le llamaría José Antonio", "Llegó para ver el partido del Viernes por la tarde", "Dí un caramelo a los niños", ...***
  - **El lenguaje natural está lleno de ambigüedades, aunque la mayoría de las veces las resolvemos inconscientemente**
- **Tipos de ambigüedades:**
  - **Léxica: la misma palabra con distintos significados (*Ej: chino, radio, planta,...*)**
  - **Sintáctica: la misma frase se corresponde con distintas estrúcturas sintácticas**
  - **Semántica: la misma frase condistintos significadas (en la mayoría de los casos, proveniente de ambigüedades léxicas o sintácticas, aunque no necesariamente)**

Eliminación de ambigüedades

- El proceso de análisis semántico puede proporcionar más de una interpretación formal de una frase
- La eliminación de ambigüedades consiste en elegir cuál de esas interpretaciones se corresponde más con la intención del comunicante
  - Se trata de asociar una medida a cada posible interpretación y escoger la de mayor valoración
- Posibles medidas:
  - Heurísticas simples: cercanía de los complementos
  - Heurísticas sofisticadas: razonamiento probabilístico
- Técnicas probabilísticas en el procesamiento del lenguaje natural
  - Gramáticas probabilísticas

Aplicaciones del procesamiento de lenguaje natural

- **Aplicaciones clásicas:**
  - **Traducción automática de textos**
  - **Extracción de información**
  - **Interfaces en lenguaje natural para: acceso a bases de datos, consulta telefónica, tutores inteligentes,...**
- **Aplicaciones actuales que tienen alguna componente PLN:**
  - **Recuperación de información, buscadores en la web**
  - **Minería de textos**
  - **Traducción probabilística**
- **Nuevas tendencias en PLN**
  - **Basado en grandes *corpus* textuales más que en frases individuales**
  - **Combinación de modelos lingüísticos y estadísticos**
  - **Los modelos probabilísticos se *aprenden* usando técnicas de aprendizaje automático**

Una aplicación: razonamiento y lenguaje natural

- **A continuación vemos con detalle una aplicación concreta: mantenimiento y consultas de una base de conocimiento**
  - **El conocimiento se añade y se consulta en lenguaje natural**
  - **La respuesta a una consulta se da en lenguaje natural y puede implicar *deducir* información a partir de lo afirmado anteriormente**
  - **En lugar de traducir a lógica de primer orden, *traducimos a cláusulas Prolog***
  - **Esto nos permite combinar el procesamiento del lenguaje natural con el motor de inferencia de Prolog**
- **Ejemplo de asertos:**
  - ***Juan es andaluz*: asertar en Prolog el hecho `andaluz(juan).`**
  - ***Todo andaluz es europeo*: asertar en Prolog la cláusula `europeo(X) :- andaluz(X).`**
- **Ejemplo de consultas:**
  - ***¿Es Juan europeo?*: consultar a Prolog el objetivo `?- europeo(juan).`**
  - ***¿Quién es europeo?*: consultar a Prolog el objetivo `?- europeo(X).`**

Aplicación: razonamiento y lenguaje natural (sesión)

- **Sesión con adición de información y con consultas:**

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

Yes
```

Aplicación: razonamiento y lenguaje natural (gramática)

- **GCD para definir e interpretar el lenguaje del usuario:**

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

sintagma_verbal(X,A)   --> verbo_cop, adjetivo(X,A).

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

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

Aplicación: razonamiento y lenguaje natural (gestor, I)

- **Programa Prolog que gestiona el intercambio de información:**

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

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

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]).
```

Aplicación: razonamiento y lenguaje natural (gestor, II)

```
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).
```

Aplicación: razonamiento y lenguaje natural (gestor, III)

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

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, 3rd ed.* (Addison-Wesley, 2001).
  - Cap. 21: "Language Processing with Grammar Rules".
- Escolano, F. y otros *Inteligencia artificial: modelos, técnicas y áreas de aplicación* (Thomson, 2003).
  - Cap. 6: "Lenguaje Natural".
- Russel, S. y Norvig, P. *Inteligencia artificial: Un enfoque moderno* (Prentice-Hall, 1996).
  - Cap. 22: "Agentes que se comunican".
- Russel, S. y Norvig, P. *Artificial Intelligence: A Modern Approach (2nd edition)* (Prentice-Hall, 2003).
  - Cap. 22: "Communication".