Tratamiento del lenguaje natural

- **Tema 5: Tratamiento del lenguaje natural**
  - Comprensión de LN
    - Análisis morfo-léxico
      - Diccionarios y reglas morfológicas
    - Análisis sintáctico
      - ATNs
      - DCGs
    - Análisis semántico
      - Gramáticas semánticas
      - Gramáticas de casos
    - Análisis pragmático
  - Generación de LN
  - Recuperación de información

Tratamiento del lenguaje natural

- El lenguaje natural es una forma de comunicación *imprecisa* y *ambigua* que se apoya en el conocimiento *compartido* por los que se comunican
  - Además, el lenguaje natural está en continua expansión y permite expresar una misma idea de muchas formas
- Tratamiento del lenguaje natural
  - Comprensión
  - Generación
- Aplicaciones
  - Traducción automática
  - Interfaces de usuario
  - Acceso a bases de datos
  - Búsqueda y filtrado de información, etcétera
- Se considera sólo el lenguaje escrito
  - No se contempla el lenguaje oral: fonología, ruidos, entonación, ambigüedades de la señal sonora *(ver Rich & Knight)*

Comprensión del lenguaje natural

- Gran dificultad debido a que el lenguaje es algo vivo: en continua expansión, que va modificándose...
  - El lenguaje se modifica tanto en vocabulario como en sintaxis
- Existencia de jergas locales, profesionales, por franjas de edad...

- Ambigüedad
  - Léxica: *Entró en el banco. Se sentó en el banco.*
    - Polisemia: ambigüedad de las palabras
  - Sintáctica: *Juan vio a María con unos prismáticos.*
    - A veces es imposible de solucionar
  - Semántica: *Los niños compraron el libro de Peter Pan.*
  - Referencial: *El jamón está en el armario. Sácalo. Ciérralo.*

- Requiere mucho conocimiento: objetivos del hablante, hipótesis, contexto... No es mera transmisión de palabras...

Niveles de análisis del lenguaje

- Fonética
  - Combinación de sonidos (fonemas) que producen palabras
- Morfología
  - Estructura de las palabras (morfemas): género, número...
- Sintaxis
  - Combinación de palabras que producen frases
- Semántica
  - Significado de palabras + estructura de frase: significado de frase (normalmente, independientemente del contexto)
- Pragmática
  - Significado de frase + contexto: significado más profundo
- Conocimiento del discurso y del mundo
  - Suele ser esencial disponer de conocimiento no explícito

Proceso de comprensión

- Comprensión: proceso de correspondencia de una forma de entrada a otra representación más útil para una cierta tarea
- Suele requerir 4 fases
  - Estas fases suelen tener límites difusos
  - No todos los sistemas trabajan con las 4 fases necesariamente
  - Análisis morfo-léxico
    - El texto se separa en unidades léxicas con significado (lexemas)
      - Pueden ser fragmentos de una palabra, una o varias palabras
    - Se obtienen los lexemas y morfemas gramaticales
    - Se les asignan una o varias categorías sintácticas (si hay ambigüedad)
  - Análisis sintáctico
    - Se comprueba la corrección de una frase con respecto a las reglas del lenguaje (gramática)
    - Se convierte la secuencia (lineal) de unidades léxicas en estructuras sintácticas (no lineales)

Proceso de comprensión

- Análisis semántico
  - Se asigna significado a una frase
  - Correspondencia entre estructuras sintácticas y objetos del dominio representados en algún lenguaje de representación del conocimiento
    - Se descartan las estructuras sintácticas para las que no pueda hacerse esta correspondencia
- Análisis pragmático
  - Hasta esta fase el análisis se hace a nivel de frase. Aquí, pasan a considerarse párrafos enteros o incluso textos de forma global
  - La mayoría de los sistemas reales no la incluyen
    - Líneas de investigación abiertas
  - Se resuelven las referencias relativas al contexto (ambigüedades referenciales, como por ej. referentes de un pronombre) y se asigna significado global a un conjunto de frases

Proceso de comprensión

- Dependiendo de las aplicaciones puede no ser necesario realizar todas las fases
- Tampoco es necesario el desarrollo secuencial de las fases
  - Puede ser necesario hacer alguna vuelta atrás
  - Propagar de una fase a la siguiente todas las alternativas: algunas se descartarán
  - Hay casos en los que se realiza un análisis sintáctico seguido de un análisis semántico totalmente independientes
  - En otros casos se realiza un análisis sintáctico-semántico en una sola etapa, dirigido por la sintaxis
  - En algunos sistemas se realiza un análisis semántico y la sintaxis juega un papel secundario
  - También se puede utilizar una arquitectura de pizarra (memoria compartida entre los distintos módulos). En cada momento actúa sobre la pizarra el módulo de análisis más conveniente

Análisis morfo-léxico

- Se basa en
  - Diccionarios (lexicones)
  - Reglas morfológicas (cómo se forman las derivaciones de palabras)
- Ambas cosas son interdependientes
  - Si en el diccionario sólo guardamos lexemas, necesitaremos muchas reglas morfológicas
  - Si guardáramos todas las formas de las palabras en el diccionario, no necesitaríamos reglas morfológicas
  - Hay que llegar a un compromiso entre ambos extremos
- Dificultades
  - Polisemia: palabra con varios significados
    - Ej.: banco (dinero, sentarse, río, peces)
  - Homonimia: palabras distintas con la misma grafía
    - Ej.: divorciado (nombre, adjetivo y verbo)

Diccionarios

- En los diccionarios se guardan distintos tipos de información:
  - Categoría sintáctica
    - Categorías cerradas (preposiciones, conjunciones, etc.)
    - Categorías abiertas (nombre, adjetivo, verbo)
    - Si una palabra tiene varias categorías, tiene que aparecer para cada una de ellas
  - Concordancia
    - Género, número, persona, caso
  - Preposiciones que admite un verbo, tipos de complementos, etc.
  - Información morfológica (patrón de formación de la palabra)
  - Información semántica
    - Concepto correspondiente, palabras sinónimas
    - Comprensión:
      - Identificar con qué concepto se corresponde la palabra
      - Varias acepciones → varios conceptos

Diccionarios

- Los diccionarios se suelen organizar utilizando relaciones de herencia múltiple, tanto de tipo gramatical como conceptual
- La construcción se realiza de forma manual o semiautomática a partir de una ontología de conceptos bajo la cual se van colocando las palabras
- Se implementan con tablas hash, tries o árboles B
  - Tries: árboles para la recuperación de palabras *(re / trie / val)*

Fases del análisis morfo-léxico

- Suele incluir 4 etapas:
  - Descomposición del texto en palabras y signos de puntuación
  - Análisis morfológico: descomponer una palabra en su raíz (lexema), prefijos y sufijos, siguiendo 3 tipos de reglas:
    - Flexión: plural, femenino, conjugaciones de los verbos, declinaciones según el caso *(yo, me)*
    - Derivación: a partir de otra de diferente categoría *(adjetivo → adverbio)*
    - Composición *(limpia/parabrisas)*
  - Búsqueda en diccionario: anterior o posterior al análisis morfológico
  - Tratamiento de errores
    - Si no se encuentra en el diccionario, a veces se puede establecer su categoría por la terminación *("mente" → adverbio)*
    - Si empieza por mayúscula se puede considerar nombre propio
    - Puede haber formatos especiales (fechas, DNI,...)
    - *Verificadores ortográficos*. Palabras que se diferencian en una letra o con letras intercambiadas

WORDNET *(Princeton University)*

- Base de datos léxica del inglés organizada semánticamente
  - http://wordnet.princeton.edu/
- La 1ª versión (1993) contenía 95.600 palabras o grupos de palabras organizadas en 70.100 significados diferentes
  - (la actual: 147.278 y 117.659)
- Relación entre palabras por distintas relaciones semánticas:
  - sinonimia, antonimia
  - es-un: hiponimia, hipernimia *(inversa de es-un)*
  - parte-de: meronimia, holonimia *(inversa de parte-de)*
- Organización de palabras en una jerarquía de unos 12 niveles que permite la herencia (basada en ideas de redes semánticas)
  - Hay 4 categorías sintácticas: nombre, verbo, adjetivo y adverbio
  - Hay palabras que están repetidas en más de una categoría sintáctica (homonimia)
  - Una palabra puede tener varios significados (polisemia) y para cada uno de ellos se da una lista de sinónimos (*synset: los conceptos*)

WordNet

- Elementos principales de WordNet
  - Conceptos = conjuntos de palabras sinónimas (*synset*)

    car $\in$ {car, auto, automobile, machine, motorcar}
  - Relaciones importantes
    - Sinonimia-antonimia

      rapidly/speedly $\leftrightarrow$ slowly
    - Hiponimia-hipernimia (sub/super-concepto)

      {car, auto, automobile, machine, motorcar}

      isa {motor vehicle, automotive vehicle}

      isa {vehicle}
    - Meronimia (parte/miembro)

      {car, auto, automobile, machine, motorcar}

      hasa {automobile engine}

      hasa {bumper}

WordNet

- WordNet (investigation)
  - Sense 1
  - probe, probing, investigation -- (an inquiry into unfamiliar or questionable activities; "there was a congressional probe into the scandal")
    - inquiry, enquiry, research -- (a search for knowledge; "their pottery deserves more study than it has received")
  - Sense 2
  - investigation, investigating -- (the work of inquiring into something thoroughly and systematically)
    - work -- (activity directed toward making or doing something; "she checked several points needing further work")
  - http://wordnetweb.princeton.edu/perl/webwn

Análisis sintáctico

- "Deslinearización" de una frase, determinando las funciones que realizan las palabras que la componen, para obtener así una estructura jerárquica que permita asignarle significado
- El significado de una frase depende en gran medida del orden de sus palabras
- Cada lenguaje natural tiene sus propias reglas en cuanto a la aceptabilidad de las posibles combinaciones de palabras como frases válidas
  - Estas reglas se suelen representar en forma de gramáticas
- La mayoría de los sistemas realizan análisis sintáctico usando
  - Una gramática: representación declarativa de la sintaxis del lenguaje
  - Un analizador sintáctico *(parser)*: programa que compara las reglas de la gramática con las frases a analizar para producir las estructuras correspondientes

Gramáticas

- Una gramática G se define como una cuádrupla (VN, VT, P, S) donde
  - VN: vocabulario no terminal
  - VT: vocabulario terminal (disjunto del anterior)
  - P: conjunto finito de producciones (reglas de reescritura)
  - S: símbolo inicial perteneciente a VN
  - Cada producción es de la forma $X \rightarrow Y$, siendo
    - X e Y cadenas de símbolos de $V \equiv VN \cup VT$
    - X cadena no vacía

- Una gramática describe las estructuras válidas del lenguaje que denota
  - Se usan aquí para el análisis sintáctico de las frases
  - El resultado puede ser uno o varios árboles de análisis sintáctico

Tipos de gramáticas

- Jerarquía de Chomsky (1957):
  - Gramáticas tipo 0 (Recursivamente enumerables)
    - Sin restricciones en cuanto a la forma de las producciones
    - Equivalentes en potencia expresiva a las máquinas de Turing
  - Gramáticas tipo 1 (Dependientes del contexto)
    - $\alpha A \beta \rightarrow \alpha \gamma \beta$ siendo $A \in VN$, $\alpha, \beta \in V^*$, $\gamma \in V^+$
  - Gramáticas tipo 2 (Independientes del contexto)
    - $A \rightarrow \gamma$ siendo $A \in VN$, $\gamma \in V^*$
    - Equivalentes en potencia a los autómatas de pila
  - Gramáticas tipo 3 (Regulares)
    - $A \rightarrow a$ o $A \rightarrow a\ B$ siendo $A, B \in VN$, $a \in VT$
    - Formato equivalente: $A \rightarrow w$ o $A \rightarrow w\ B$ siendo $w \in VT^*$ (lineales por la derecha).
    - Equivalentes en potencia a los autómatas finitos

Gramáticas para lenguajes naturales

- Las gramáticas independientes del contexto (GICs) son muy utilizadas como gramáticas para LPs y lenguajes naturales
  - Hay construcciones de algunos lenguajes naturales que no son independientes del contexto (p.ej., los fenómenos de concordancia)
  - La mayor parte de su estructura sí lo es
- Las regulares se ajustan peor, incluso a los LPs
  - Imposibilidad de capturar anidamientos, por ejemplo
- Existen muchos mecanismos eficientes de *parsing* para este tipo de gramáticas (tipo 2 y 3)
- Lo más habitual en el procesamiento del lenguaje natural es utilizar gramáticas independientes del contexto con algún tipo de extensión
  - Estas características adicionales las dotan de mayor capacidad expresiva y cambian su lugar en la jerarquía

Gramáticas para lenguajes naturales

- El hecho de si los lenguajes naturales son o no expresables con gramáticas de tipo 2 es motivo de controversia entre los lingüistas
  - Hasta mitad de los 80, los lingüistas se concentraron en los lenguajes independientes del contexto y dependientes del contexto (tipo 2 y tipo 1)
  - Desde entonces, ha crecido la importancia de los lenguajes regulares, ocasionado por la necesidad de procesar muy rápidamente cantidades ingentes de *texto en línea*, incluso a costa de un análisis menos completo
  - Como dice Fernando Pereira:

    "Cuanto más viejo me vuelvo, más abajo voy en la jerarquía de Chomsky"

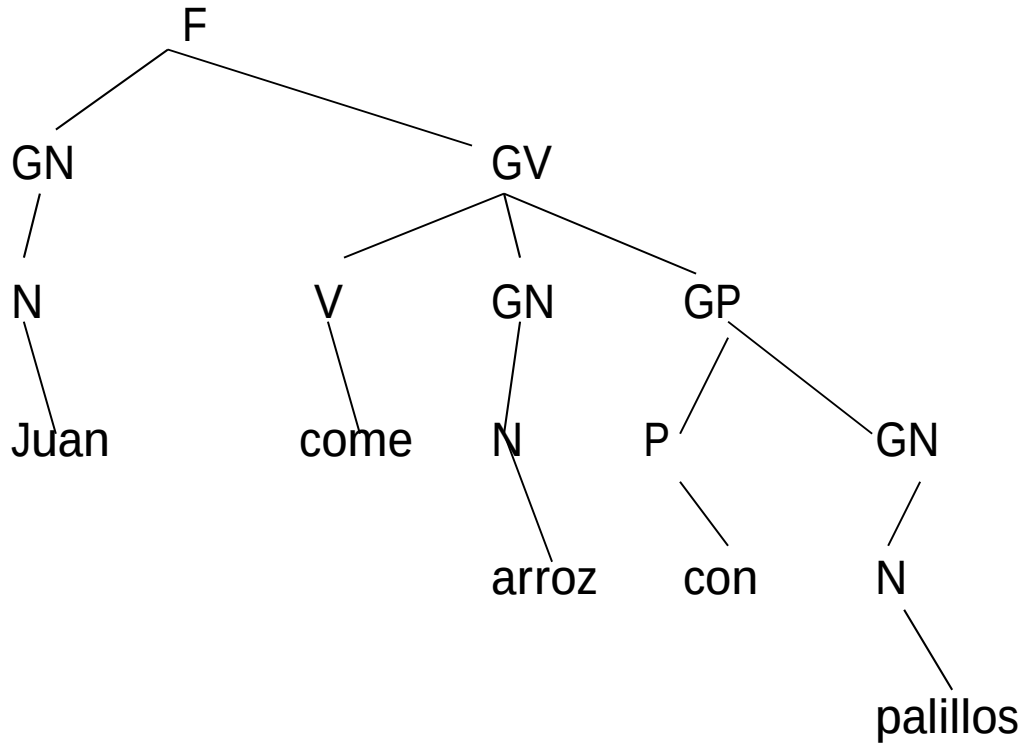
Tipos de análisis sintáctico

- Método descendente (razonamiento hacia delante)
  - Empieza a partir del símbolo inicial y va aplicando producciones hasta que se ha desarrollado el árbol de derivación y los símbolos terminales que etiquetan a las hojas se corresponden con la frase
- Método ascendente (razonamiento hacia atrás)
  - El analizador empieza por los símbolos terminales de la frase, viendo a qué categoría corresponden y, aplicando las reglas al revés, intenta alcanzar el símbolo inicial
- La elección suele depender
  - En primer lugar, del factor de ramificación
  - También de si hay heurísticas aplicables para desechar caminos
- Método híbrido: ascendente con filtro descendente
- La salida puede no ser una única estructura
  (si hay ambigüedad sintáctica y se quieren varias respuestas)

Ambigüedad sintáctica

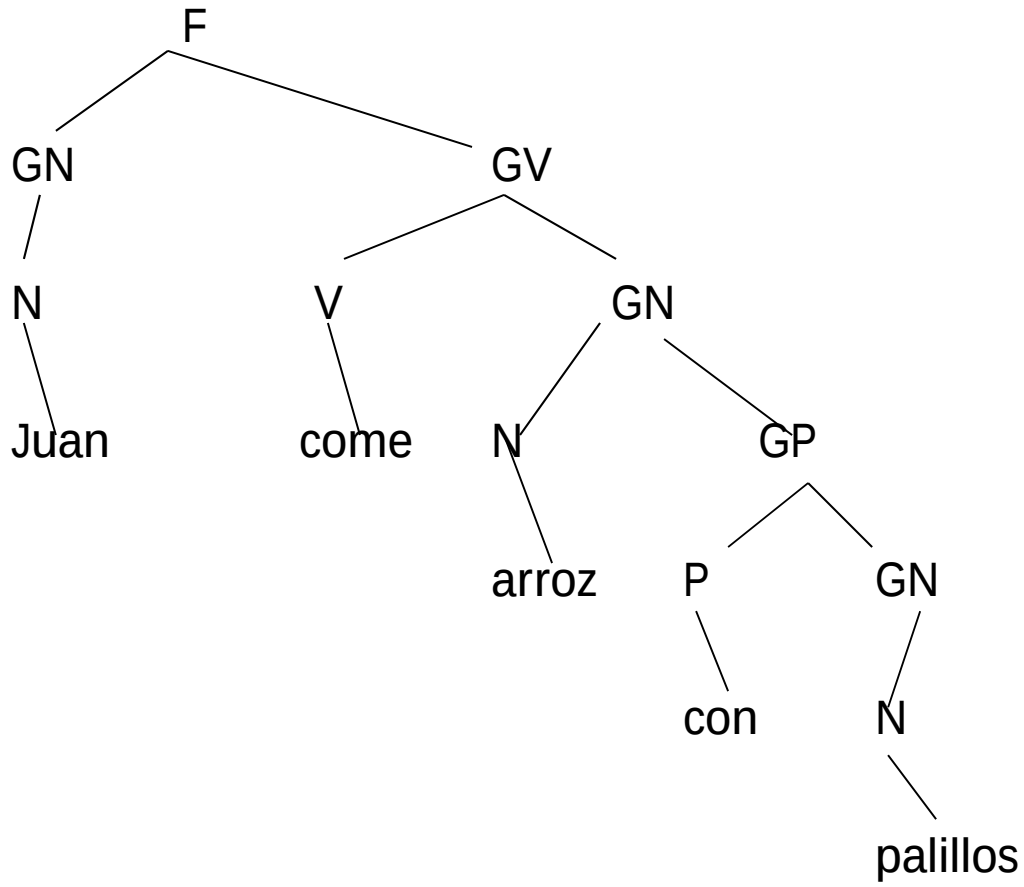
- Ejemplo
  - “Como todos los días...”
  - No se ha analizado toda la frase: ¿conjunción o verbo?
  - Podría aclararse al seguir leyendo
    - “Como todos los días llegas tarde” / “Como todos los días ensalada”
  - Hay varias alternativas para lidiar con esto en la implementación de los analizadores sintácticos
- Pero hay frases “genuinamente” ambiguas
  - “Estaba en casa cuando llamó”
  - ¿1ª o 3ª persona?
  - O se siguen todos los caminos
    - Costoso por mantener estructuras intermedias que se desecharán
  - O se contempla el backtracking
    - Costoso también por cómputos duplicados
  - Lo más habitual: una sola interpretación plausible en esta fase

Ambigüedad sintáctica



- Este análisis es correcto desde el punto de vista sintáctico. Además es válido semánticamente porque los palillos son un instrumento y no algo comestible

Ambigüedad sintáctica



- Este análisis es correcto desde el punto de vista sintáctico, pero no desde el punto de vista semántico. Lo sería para una frase como “Juan come arroz con pimientos”

Análisis sintáctico

- Tipos de analizadores sintácticos más usados en sistemas de tratamiento del lenguaje natural
  - Gramáticas aumentadas
    - Extensión de las GICs (p.ej. para concordancia)

Formalismo / Parte del análisis	ATNs (redes con transiciones aumentadas)	Gramáticas de unificación
Reconocimiento (encaje)	Comprobación categorías de palabras y tests asociados a los arcos	Unificación
Construcción de la estructura de frase	Asignación	Unificación

ATNs *(Augmented Transition Networks)*

- Las ATNs se corresponden con un proceso de análisis sintáctico descendente
  - Similar a una máquina de estados finitos en la que se ha ampliado el tipo de etiquetas que se pueden asociar a los arcos que definen las transiciones
  - Los arcos se pueden etiquetar con palabras específicas, con categorías de palabras (nombre, adjetivo, ...), llamadas a otras redes, procedimientos diversos...
- El inconveniente de las ATNs es su importante componente procedimental, que limita su utilidad
  - Describe el orden en el que deben construirse los constituyentes
  - Los sistemas no pueden usar la misma gramática tanto para comprensión como para generación
  - Es el compromiso procedimental/declarativo que afecta a la flexibilidad de la representación: más declarativa, más flexible

Gramáticas de unificación

- Las gramáticas de unificación o gramáticas lógicas utilizan el mecanismo de unificación como base para el proceso de análisis
  - Este mecanismo es mucho más potente que los mecanismos de comparación y asignación utilizados en las ATNs y además no requiere que el árbol de derivación de una parte de la frase esté completado antes de empezar a analizar la siguiente
- Las gramáticas de unificación se utilizan, además de en el análisis sintáctico, en otros análisis como el morfológico y el semántico
- Las gramáticas de cláusulas definidas (DCGs) son uno de los tipos de gramáticas de unificación más utilizados
  - Forman parte del Prolog estándar
  - Se concibieron como una generalización ejecutable de las GICs

DCGs (Gramáticas de cláusulas definidas)

- Las DCGs son un formalismo de representación de gramáticas de unificación que permite representar directamente gramáticas independientes del contexto (tipo 2)
  - Se usa --> como operador para representar la → de las reglas
  - Los símbolos no terminales deben ser átomos
  - Los símbolos terminales deben ser átomos también
    - Para distinguirlos de los no terminales, una secuencia de terminales se escribe como una lista (la cadena vacía es [ ])
  - Se separan los símbolos mediante comas
  - Las reglas terminan en punto

- Las DCGs incluyen una serie de extensiones que permiten representar características dependientes del contexto y que les confieren una potencia expresiva equivalente a las máquinas de Turing

DCGs (Gramáticas de cláusulas definidas)

- Ejemplo:

```
frase --> grupo_nominal, grupo_verbal.
grupo_nominal --> nombre.
grupo_verbal --> verbo, grupo_nominal.
grupo_verbal --> verbo.
nombre --> [manzanas].
nombre --> [juan].
verbo --> [come].
```

podría estar definiendo a una GIC G = (VN, VT, P, S) donde

- VN = {frase, grupo_nominal, grupo_verbal, nombre, verbo}
- VT = {manzanas, juan, come}
- S = frase
- P contiene las producciones anteriores

Ejemplo de uso

```
frase --> grupo_nominal, grupo_verbal.
grupo_nominal --> nombre.
grupo_verbal --> verbo, grupo_nominal.
grupo_verbal --> verbo.
nombre --> [manzanas].
nombre --> [juan].
verbo --> [come].
```

- Para analizar una frase completa, escribiríamos

```
?- frase([juan, come, manzanas], []).
```

- Es un reconocedor y no un analizador (porque no devuelve la estructura sintáctica)
- En esta DCG no está representada la concordancia
- También reconoce la frase “manzanas come Juan”

Traducción a Prolog de DCGs: idea básica

- Cada símbolo no terminal se convierte en un predicado unario cuyo parámetro representa una cadena que puede derivarse a partir de él
  - Las cadenas de símbolos terminales se representan como listas, y la concatenación de cadenas se representa como la concatenación de listas

```
% frase --> grupo_nominal, grupo_verbal.
frase(Xs) :-
        grupo_nominal(As), grupo_verbal(Bs),
        append(As, Bs, Xs).
% grupo_nominal --> nombre.
grupo_nominal(Xs) :- nombre(Xs).
% nombre --> [manzanas].
nombre([manzanas]).
```

DCGs a Prolog: listas diferencia

- Las estructuras diferencia o estructuras incompletas son una técnica de representación de datos exclusiva de Prolog, que explota el potencial de la unificación y de las variables lógicas, usándolas como *huecos* que denotan una parte de la estructura aún no computada
  - Generalizan la idea de acumulador extendiéndola a estructuras de datos arbitrarias
  - Las estructuras de datos incompletas más usadas son las listas diferencia
  - Representación de una lista como la diferencia entre dos listas
    - Resumiendo: [1, 2, 3 | Xs] \ Xs es [1,2,3] y Zs\Zs es [ ]
  - Permite concatenación en tiempo constante (de listas diferencia compatibles) usando variables como punteros

```
append_ld(Xs\Ys, Ys\Zs, Xs\Zs).    %con functor
append_ld(Xs, Ys, Ys, Zs, Xs, Zs). %sin functor
```

Traducción a Prolog de DCGs: realidad

- Los preprocesadores de DCGs que realizan la traducción de estas gramáticas a Prolog utilizan listas diferencia por eficiencia
  - Una lista diferencia As\Bs se representa con 2 parámetros As y Bs

```
% frase --> grupo_nominal, grupo_verbal.
% frase(Xs) :-
%         grupo_nominal(As), grupo_verbal(Bs),
%         append(As, Bs, Xs).
frase(A, B) :-
         grupo_nominal(A, C), grupo_verbal(C, B).
% nombre --> [manzanas].
% nombre([manzanas]).
nombre([manzanas | A], A).  % SWI: incorrecta
nombre(A, B) :- `C`(A, manzanas, B).  % correcta
% Def. de conecta `C`: `C`([X|Xs], X, Xs).
```

DCGs

- El uso de DCGs no requiere conocer su traducción a Prolog
  - Siempre podemos “cotillear” el proceso de traducción automático a cláusulas Prolog usando el predicado `listing/0`
  - Lo importante es saber que para hacer consultas hemos de añadir 2 parámetros l.d. extra al final (de los cuales, el último siempre es [ ])
  - En el ejemplo anterior, comprobamos si una frase es correcta:

```
?- frase([juan, come, manzanas], []).
```

  - O generamos todas las frases “correctas” *(derivables)*:

```
?- frase(X, []).
X= [manzanas, come, manzanas] ;
X= [manzanas, come, juan] ; X= [manzanas, come] ;
X= [juan, come, manzanas] ; X= [juan, come, juan] ;
X= [juan, come] ;
No
```

DCGs: extensiones con respecto a las GICs

- Las extensiones a las GICs que incorporan las DCGs suponen añadirles una serie de características que pueden resumirse en
  - Posibilidad de efectuar llamadas a predicados Prolog (encerrándolos entre llaves { })
  - Posibilidad de añadir parámetros a los símbolos no terminales
  - Potencia de la unificación
- Esto aumenta su utilidad y potencia expresiva, convirtiendo a las GICs en una nueva clase de gramáticas ejecutables: las DCGs
  - Permiten representar restricciones de concordancia, obtener el árbol de análisis y facilitar otras tareas relacionadas con el análisis
- Las ideas para la traducción automática a programas Prolog son las mismas
  - Los símbolos no terminales, que pueden tener ahora aridad *n*, se convierten en predicados *n+2*-arios
    - Los 2 parámetros extra son la lista diferencia y siempre son los últimos

Parámetros en símbolos no terminales

- Ejemplo de uso para garantizar la concordancia entre sujeto y verbo

```
frase --> grupo_nominal(Numero),
          grupo_verbal(Numero).

grupo_nominal(Numero) --> nombre(Numero).

grupo_verbal(Numero) --> verbo(Numero),
                         grupo_nominal(Numero1).
  %(D): no concuerda con el verbo

grupo_verbal(Numero) --> verbo(Numero).

nombre(plural) --> [manzanas].

nombre(singular) --> [juan].

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

- Mismo uso: `?- frase([juan, come, manzanas], []).`

Parámetros en símbolos no terminales

```
frase --> grupo_nominal(Num), grupo_verbal(Num).
grupo_nominal(Numero) --> nombre(Numero).
grupo_verbal(Numero) --> verbo(Numero),
                         grupo_nominal(Numero1). %(1)
grupo_verbal(Numero) --> verbo(Numero).
nombre(plural) --> [manzanas].
nombre(singular) --> [juan].
verbo(singular) --> [come].
```

- Uso para generación

```
?- frase(Xs, []).
Xs = [juan, come, manzanas] ;
Xs = [juan, come, juan] ;
Xs = [juan, come] ;
No
```

Parámetros en símbolos no terminales

- Ejemplo de uso para obtención del árbol de análisis

```
frase(f(GN, GV)) --> grupo_nominal(GN),
                     grupo_verbal(GV).

grupo_nominal(gn(N)) --> nombre(N).

grupo_verbal(gv(V, GN)) --> verbo(V),
                            grupo_nominal(GN).

grupo_verbal(gv(V)) --> verbo(V).

nombre(n(manzanas)) --> [manzanas].

nombre(n(juan)) --> [juan].

verbo(v(come)) --> [come].
```

- Uso para obtener el árbol de derivación de una frase

```
?-frase(Arbol, [juan, come, manzanas], []).

Arbol= f(gn(n(juan)),gv(v(come),gn(n(manzanas))))
```

Parámetros en símbolos no terminales

- Generación de frases y árboles

```
?- frase(Arbol, Xs, []).
Arbol = f(gn(n(manzanas)),gv(v(come),gn(n(manzanas))))
Xs = [manzanas, come, manzanas] ;
Arbol = f(gn(n(manzanas)), gv(v(come), gn(n(juan))))
Xs = [manzanas, come, juan] ;
Arbol = f(gn(n(manzanas)), gv(v(come)))
Xs = [manzanas, come] ;
Arbol = f(gn(n(juan)), gv(v(come), gn(n(manzanas))))
Xs = [juan, come, manzanas] ;
Arbol = f(gn(n(juan)), gv(v(come), gn(n(juan))))
Xs = [juan, come, juan] ;
Arbol = f(gn(n(juan)), gv(v(come)))
Xs = [juan, come] ;
No
```

Llamadas a predicados

- Otra extensión consiste en incorporar requisitos de satisfacción de predicados Prolog en la parte derecha de las reglas
- Estos predicados actúan como “llamadas a procedimientos externos a la gramática”
- Deben distinguirse sintácticamente para evitar que sean considerados como categorías sintácticas por el preprocesador de DCGs
  - Si no, les añadiría los dos argumentos habituales, y probablemente resultaría en un error de ejecución
  - Por ello las llamadas a predicados Prolog se colocan entre llaves en las reglas
  - Su traducción a Prolog es directa: se mantienen tal cual (eliminando las llaves)

Llamadas a predicados

- Ejemplo de uso en construcción de diccionarios:
  - En vez de

```
nombre(n(manzanas)) --> [manzanas].
nombre(n(juan)) --> [juan].
```

  - Conviene poner

```
nombre(n(P)) --> [P], {es_nombre(P)}.

es_nombre(manzanas).
es_nombre(juan).
```

- De esta forma, la implementación del diccionario se hace en forma de hechos y no mediante reglas que harían crecer innecesariamente el tamaño de la gramática
  - No era razonable tener una regla por cada palabra

Llamadas a predicados

- Ejemplo de uso para comprobación de preposiciones permitidas por verbos *(sub-categorización de verbos)*

```
frase --> nombre_propio, verbo(X), complemento(X).
complemento([]) --> [].
complemento([X]) --> preposición(X), nombre_propio.
verbo(C) --> [P], {es_verbo(P, C)}.
nombre_propio --> [P], {es_nombre_p(P)}.
preposición(P) --> [P], {es_prepo(P)}.
```

```
es_verbo(piensa, [en]).      es_nombre_p(juan).     es_prepo(en).
es_verbo(ríe, []).           es_nombre_p(maría).    es_prepo(con).
es_verbo(habla, [con]).      es_nombre_p(ana).
```

- ¿Qué tipo de frases permitiríamos reconocer con esta gramática?

Llamadas a predicados

- Generación de las frases que reconocería la DCG anterior con

```
?- frase(Xs, []).
```

- Las siguientes combinaciones se reconocerían como correctas
  - Juan/María/Ana piensa en Juan/María/Ana
  - Juan/María/Ana ríe
  - Juan/María/Ana habla con Juan/María/Ana

Llamadas a predicados

- Si queremos que haya más de un complemento:

```
complemento([]) --> [].
complemento([X|Xs]) --> preposición(X),
                        nombre_propio,
                        complemento(Xs).    % cambio

es_verbo(habla, [de, con]).      % cambio
es_prepo(de).                    % se añade
```

  - Siempre: "... habla de ... con ..."

- Si se quiere que haya más flexibilidad en el orden de los complementos y que algunos puedan ser opcionales, hay que recurrir a un tratamiento de listas más elaborado, utilizando por ejemplo el predicado `member/2`