Pontificia Universidad Católica de Chile

Facultad de Letras

Introducción a la lingüística computacional

César Antonio Aguilar
Facultad de Lenguas y Letras
16/10/2017

Cesar.Aguilar72@gmail.com

Gramáticas léxicas (1)

En esta clase vamos a seguir revisando modelos de gramáticas formales, y ahora nos vamos a enfocar en aquellas que tratan de relacionar estructuras sintácticas con información semántica.

Los modelos que vamos a ver son **Lexical-Functional Grammar** (LFG) y **Head-Driven Phrase Structure Grammar** (**HPSG**).

LFG

Lexical-Functional Grammar

[HEAD-DRIVEN GRAMMAR [PHRASE STRUCTURE]]

HPSG

Gramáticas léxicas (2)

A diferencia de TAG, estas gramáticas intentan ligar un mínimo de información semántica a las palabras que configuran una estructura sintáctica, con el fin de optimizar la comprensión y producción de tales estructuras.

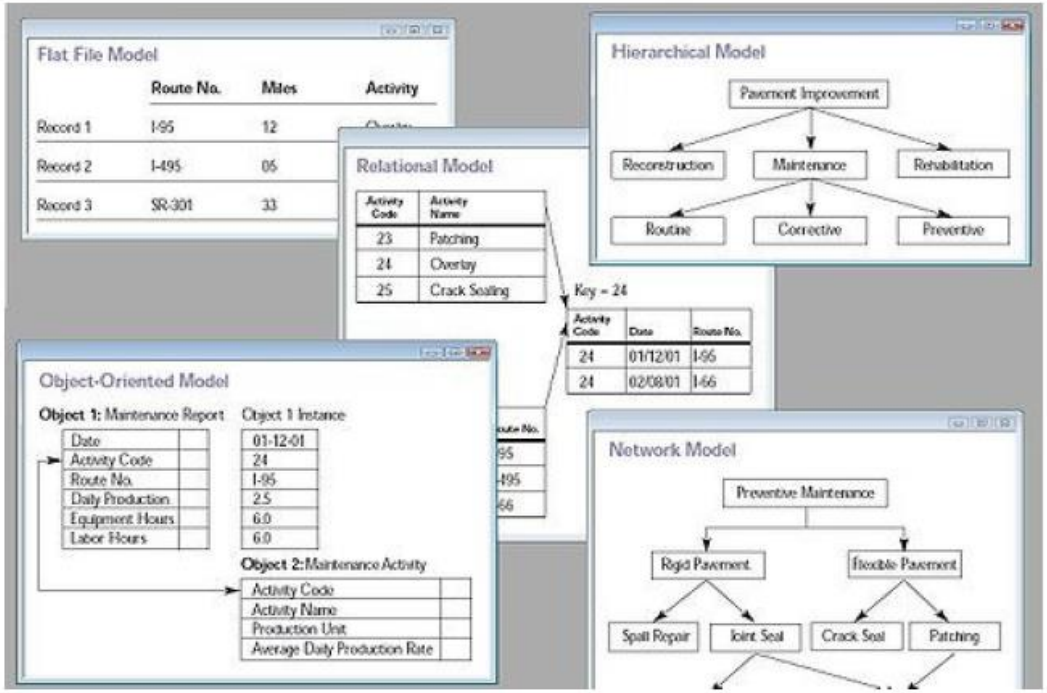
A estas gramáticas se les conoce como *gramáticas basadas en rasgos*, y han sido propuestas y trabajadas por muchas personas.

$$\left\langle \text{amar}, \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \text{verb} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & \left\langle \begin{matrix} \text{NP} \\ [\text{INDEX } i] \end{matrix} \right\rangle \\ \text{COMPS} & \left\langle \begin{matrix} \text{NP} \\ [\text{INDEX } j] \end{matrix} \right\rangle \end{bmatrix} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{MODE} & \text{prop} \\ \text{INDEX} & s \\ \text{RESTR} & \left\langle \begin{bmatrix} \text{RELN} & \text{amar} \\ \text{SIT} & s \\ \text{AMANTE} & i \\ \text{AMADO} & j \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix} \right\rangle$$

Gramáticas léxicas (3)

Grosso modo, una gramática de rasgos explica que para generar una oración, se necesitan dos elementos básicos:

- Un lexicón que contenga toda la información necesaria para construir una buena oración.
- Un conjunto de reglas de construcción y combinación que establezca cuál es el patrón a seguir para construir una buena oración.



Sin querer ser muy computacionales, un lexicón lo podemos ver como una enorme base de datos que relaciona objetos (palabras) con atributos (o ragos)

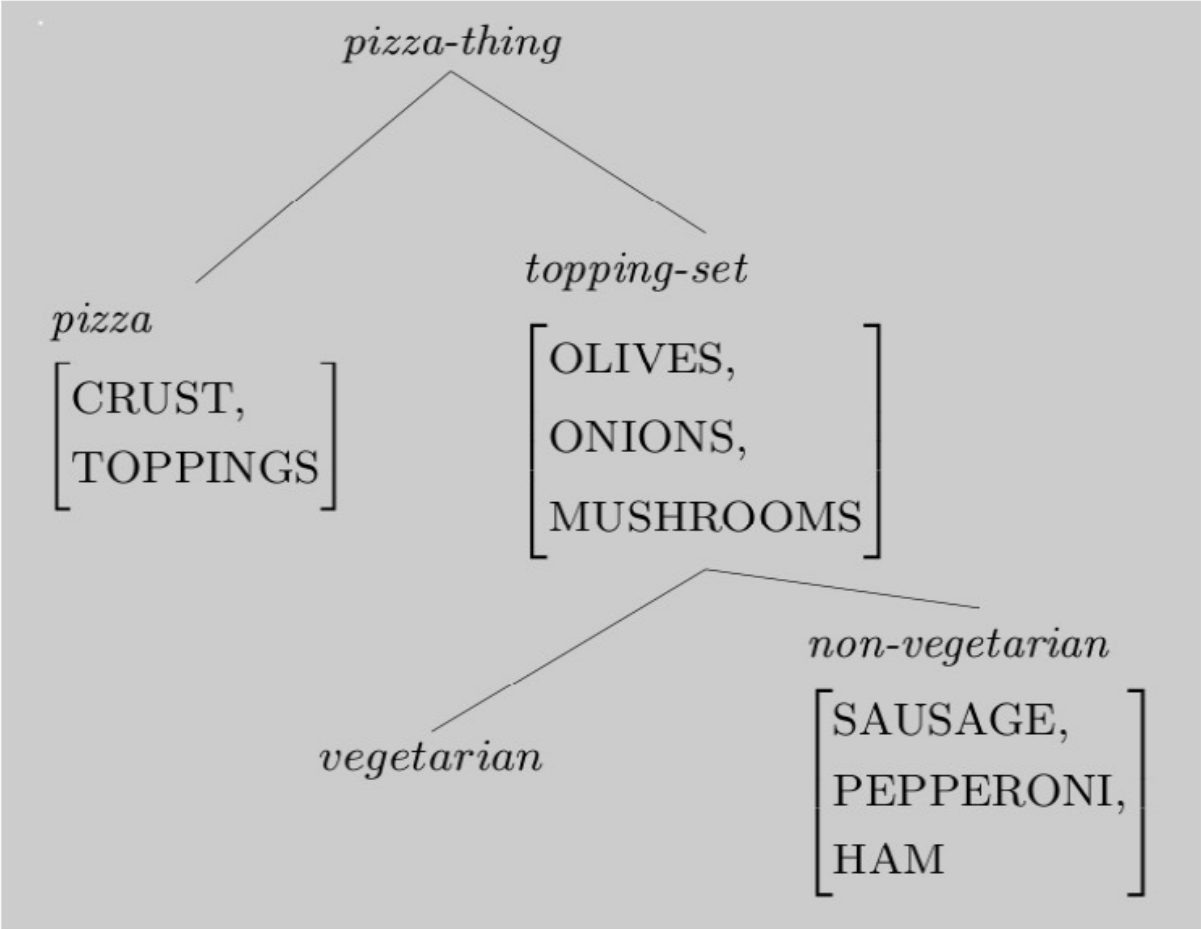
Gramáticas léxicas (4)

El mecanismo es muy simple: supongamos que queremos clasificar objetos que tienen rasgos particulares, los cuales les dan valores concretos a nuestros objetos, esto es:

$$\begin{bmatrix} \text{FEATURE}_1 & \text{VALUE}_1 \\ \text{FEATURE}_2 & \text{VALUE}_2 \\ \cdots & \\ \text{FEATURE}_n & \text{VALUE}_n \end{bmatrix}$$

Gramáticas léxicas (5)

Hagamos un poco de lingüística en la cocina: ¿cómo jerarquizarían un menú de pizzas?



Gramáticas léxicas (6)

Le agregamos un poco de organización a nuestra clasificación...

TYPE	FEATURES/VALUES	IST
pizza-thing		
pizza	$\begin{bmatrix} \text{CRUST} & \{\text{thick, thin, stuffed}\} \\ \text{TOPPINGS} & \textit{topping-set} \end{bmatrix}$	*pizza-thing*
topping-set	$\begin{bmatrix} \text{OLIVES} & \{+, -\} \\ \text{ONIONS} & \{+, -\} \\ \text{MUSHROOMS} & \{+, -\} \end{bmatrix}$	*pizza-thing*
vegetarian		*topping-set*
non-vegetarian	$\begin{bmatrix} \text{SAUSAGE} & \{+, -\} \\ \text{PEPPERONI} & \{+, -\} \\ \text{BBQ CHICKEN} & \{+, -\} \end{bmatrix}$	*topping-set*

Gramáticas léxicas (6)

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \text{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \textit{vegetarian} & \\ \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

{<CRUST , thick> , <TOPPINGS , { <OLIVES , + > ,
<ONIONS, +> , <MUSHROOMS, –>}>}

{<CRUST , thick> , <TOPPINGS , { <OLIVES , + > ,
<ONIONS, +> , <MUSHROOMS, +>}>}

Si pido una pizza, ¿cómo debería de hacerlo?

Gramáticas léxicas (7)

De hecho, si somos exigentes, nuestras matrices de rasgos nos permiten establecer qué se combina con qué en nuestras pizzas.

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \text{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \& \begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \text{thin} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

$$= \emptyset$$

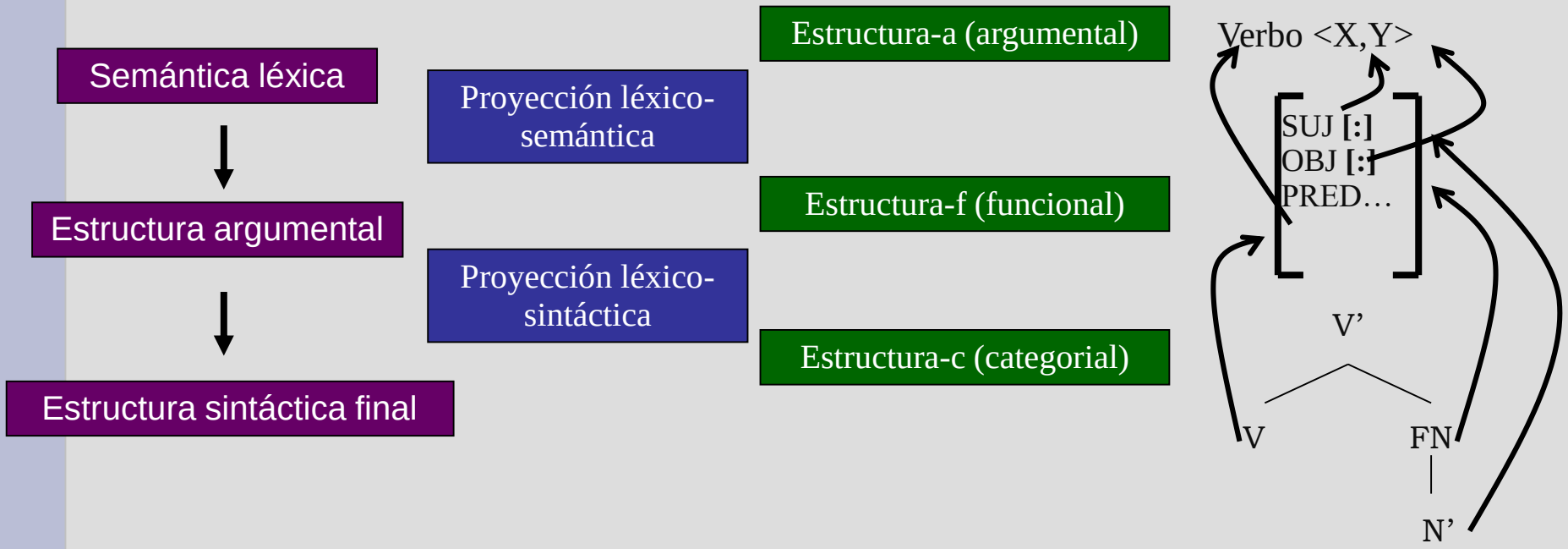
Gramáticas léxicas (8)

Cada nueva pizza que cocine tiene cabida en mi jerarquía: lo único que requiero es *indexarla* asociándole los rasgos y valores que considere pertinentes.

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \text{thin} \\ \text{ONE-HALF} & \begin{bmatrix} \text{OLIVES} & \boxed{1} \\ \text{ONIONS} & \boxed{2} \end{bmatrix} \\ \text{OTHER-HALF} & \begin{bmatrix} \text{OLIVES} & \boxed{1} \\ \text{ONIONS} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

Lexical-Functional Grammar (1)

LFG fue delineada por Kaplan y Bresnan (1982) como un modelo que permite trazar la información semántica proyectada por una estructura sintáctica. Puede decirse que dicha información semántica es codificada en un nivel léxico, la cual es representada tanto en un plano funcional (o estructura-F), como en un nivel de categorías sintácticas (estructura-C).



Lexical-Functional Grammar (2)

Vamos a ver el siguiente ejemplo: analicemos esta oración:

(1) *The cat ate the rats.*

Esta oración muestra los tres niveles que comentamos antes:

1. Estructura sintáctica (o **C-Structure**): p.e., *the cat* es un constituyente del tipo FN inserto en(1) .

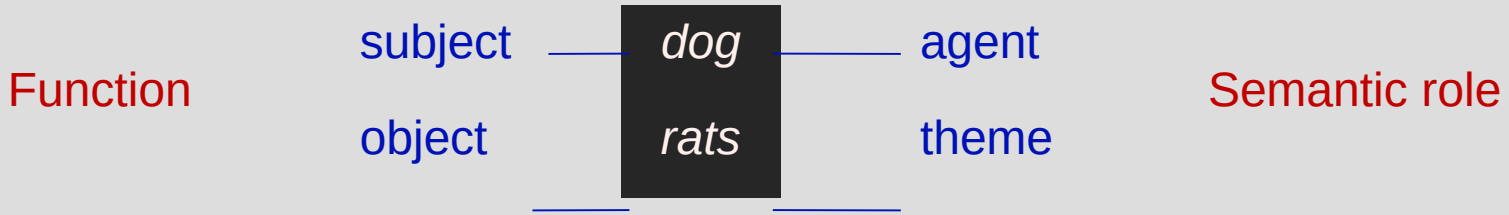
2. Estructura funcional (o **F-Structure**): p.e., *the cat* es el Sujeto de (1).

3. Estructura argumental (o **A-Structure**): p.e., *the cat*, de acuerdo con la información semántica del verbo *to eat*, es una FN que, dada su función de Sujeto, conceptualmente se desempeña como el **Agente** de esta acción, en tanto que *the rats* es el **Tema** que se ve afectado por esta acción.

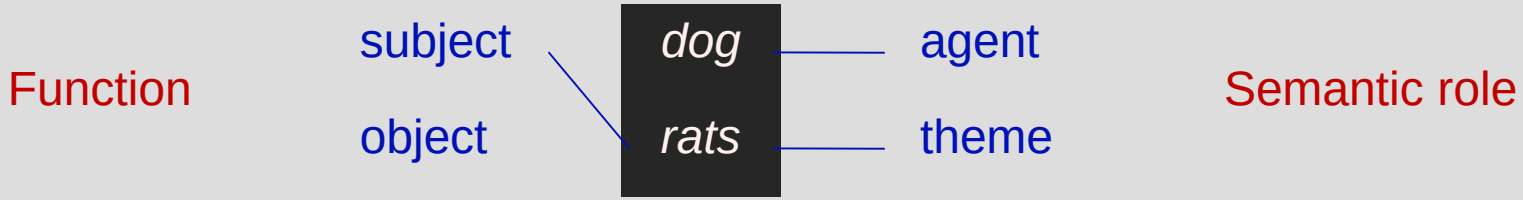
Lexical-Functional Grammar (3)

No siempre las oraciones asumen la forma de (1): también podemos hacer cambios en su estructura, p.e.:

(1) *The dog ate the rats.*

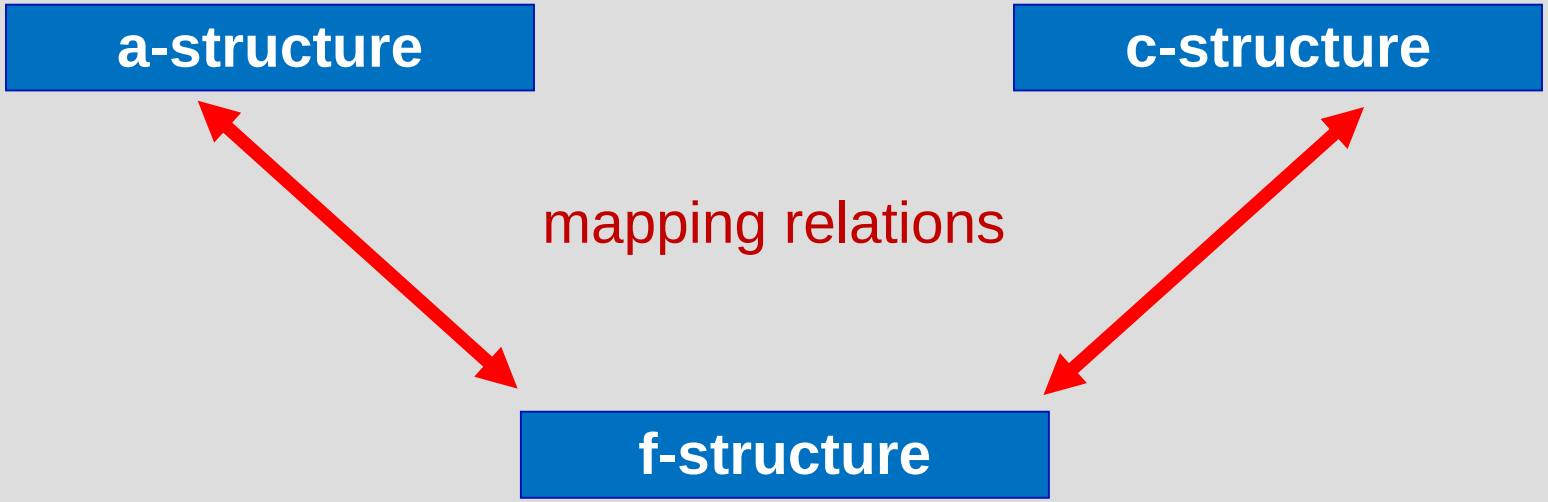


(2) *The rats were eaten by the dog.*



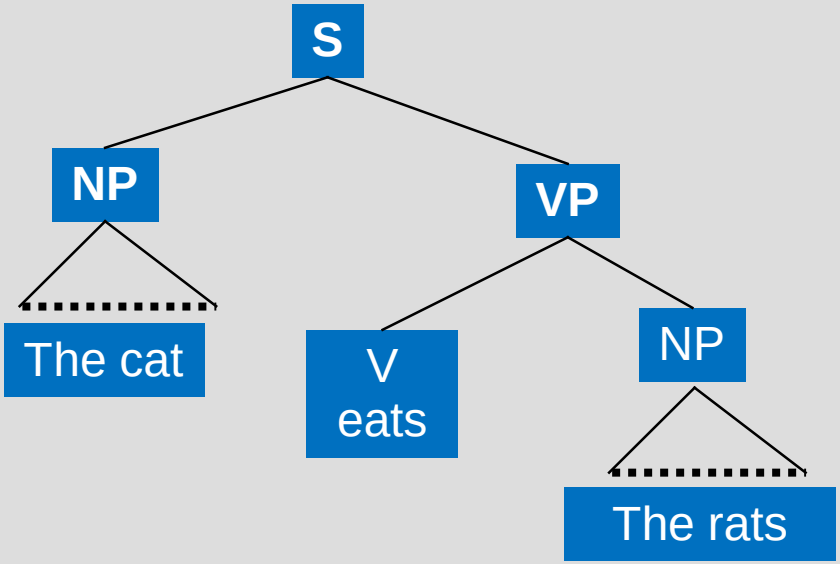
Lexical-Functional Grammar (4)

Ahora bien, estas tres estructuras mantienen relaciones entre sí, de tal suerte que (pensando en teoría de conjuntos) suponemos que existen tres conjuntos de estructuras (**C**, **F**, y **A**)´, de modo que estos conjuntos establecen **mapeos** entre sí. Veamos:



Lexical-Functional Grammar (5)

Estas relaciones de mapeo nos permiten construir **matrices de rasgos léxicos** que asociamos a cada una de las palabras de la oración que estamos analizando:



$$\begin{bmatrix} \text{PRED} & \text{'eat<SUBJ, OBJ>'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'The Cat'} \\ \text{NUM} & \text{sing} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'The Cat'} \\ \text{NUM} & \text{plur} \end{bmatrix} \\ \text{TENSE} & \textit{present} \end{bmatrix}$$

Head-Diven Phrase Structure Grammar (1)

HPSG fue planteado por Pollard y Sag (1994), y al igual que LFG intenta hacer una descripción completa de estructuras sintácticas considerando la información léxica y semántica subyacente en éstas. Considera las siguientes estructuras básicos:

a) Una estructura fonética, la cual codifica una cadena de signos en una cadena lingüística determinada.
b) Un conjunto de *categorías sintácticas*, las cuales *mapean* la estructura fonética y la asocian a determinadas categorías (N, FN, V, FV, Prep, FPrep, etc.).
c) Un conjunto de *relaciones semánticas*, las cuales ligan a las categorías anteriores algún rol léxico-semántico determinado (AGENTE, PACIENTE, INSTRUMENTAL, BENEFACTIVO, etc.

Cada uno de los objetos ubicados en estos niveles, configura dos descriptores: uno que señala los *rasgos* de cada objeto (p. e. una palabra puede tener el rasgo de Nombre, Verbo, Determinante, Adjetivo, etc.); y otro que indica los *valores* asumidos por cada rasgo o estructura de rasgos (p. e., un Verbo se caracteriza por el número de argumentos que rige; un nombre puede ser caracterizado por su género y su número, etc.).

Head-Diven Phrase Structure Grammar (2)

HPSG formula *plantillas léxicas*, las cuales indican *jerarquía de rasgos y valores*, según se puede ver a continuación:

$$\mathrm{NP} = \begin{bmatrix} phrase & \\ \mathrm{HEAD} & noun \end{bmatrix}$$

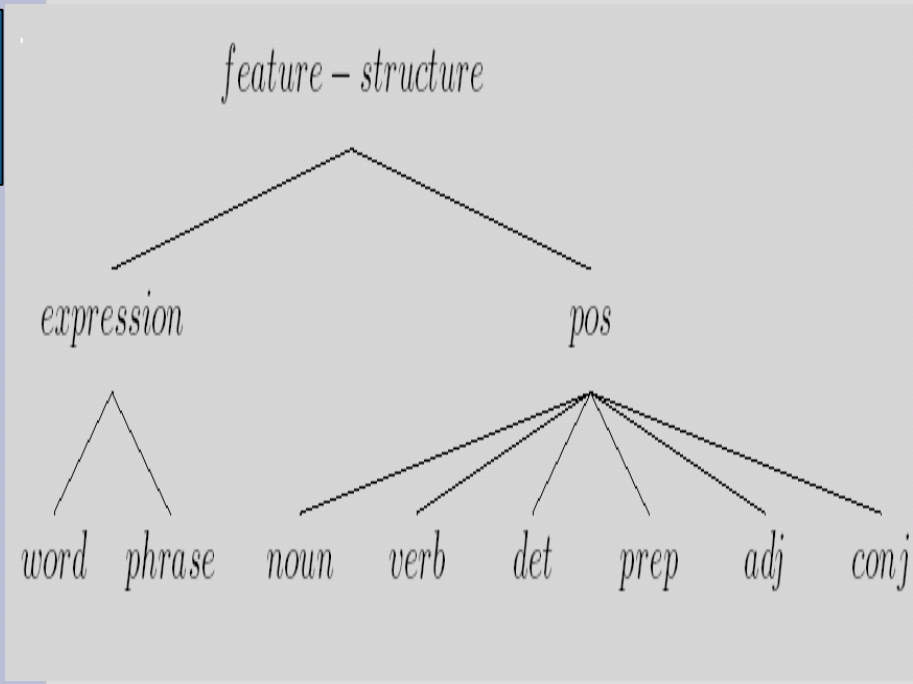
$$\mathrm{V} = \begin{bmatrix} word & \\ \mathrm{HEAD} & verb \end{bmatrix}$$

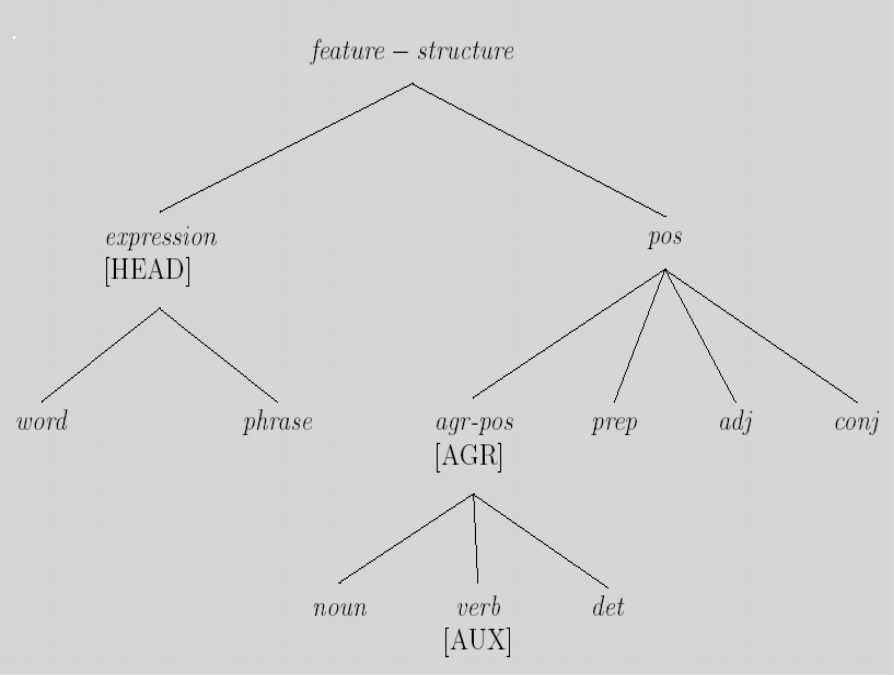
$$\left\langle \mathrm{bird}\ , \begin{bmatrix} word & \\ \mathrm{HEAD} & noun \end{bmatrix} \right\rangle$$

$$\mathrm{VP} = \begin{bmatrix} phrase & \\ \mathrm{HEAD} & verb \end{bmatrix}$$

Head-Diven Phrase Structure Grammar (3)

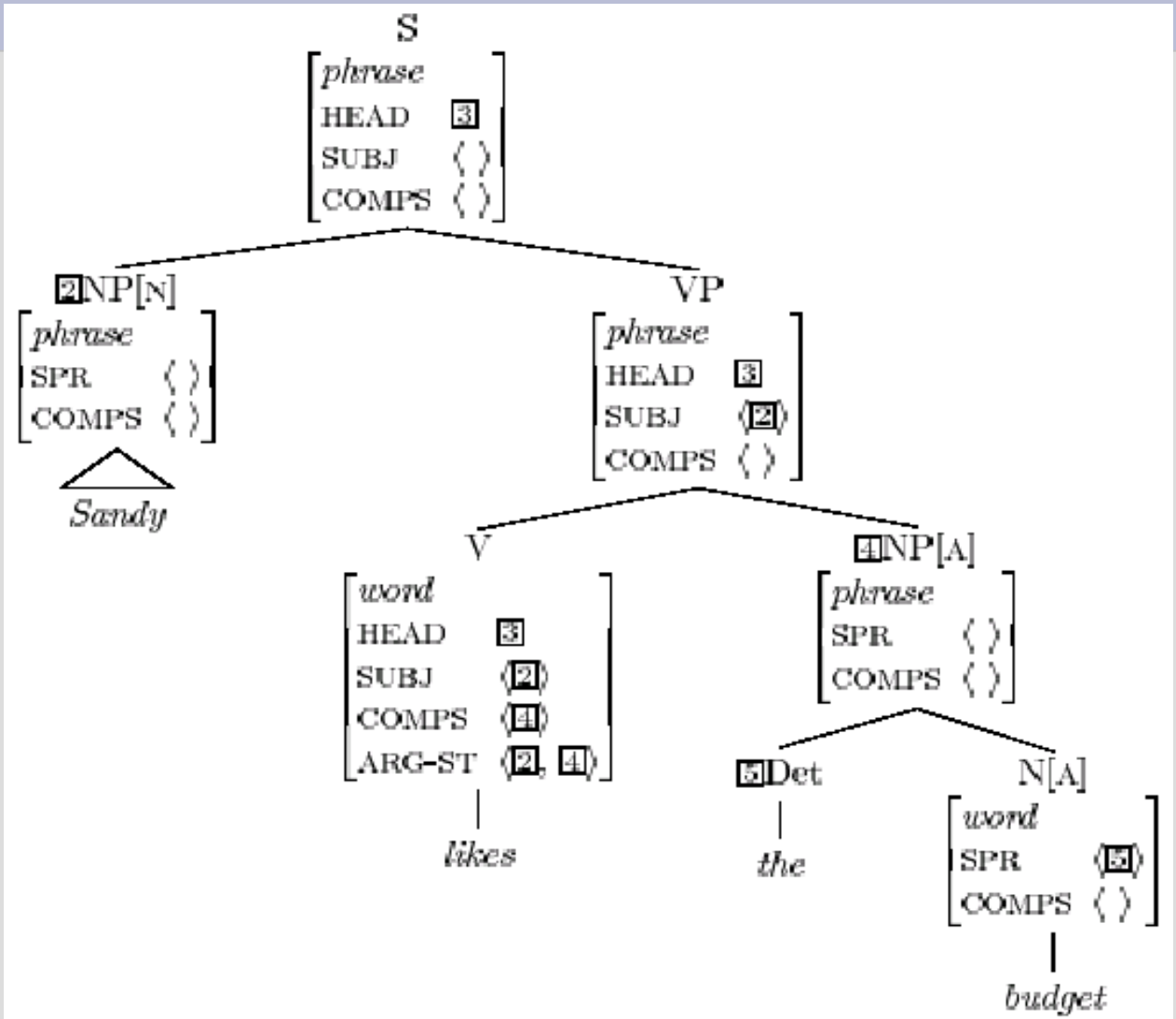
Manteniendo estas plantillas básicas, se pueden ubicar dos tipos de jerarquías para unidades constitutivas de una oración:





Head-Diven Phrase Structure Grammar (4)

Un ejemplo es: *Sandy likes the budget*



Gramáticas léxicas en NLTK (1)

Usando NLTK podemos generar nuestras propias gramáticas de rasgos, haciendo uso de un recurso que se conoce como *diccionario. Un diccionario es una estructura* de datos que nos permite asociar a un objeto un conjunto de rasgos previamente establecidos. Un ejemplo es:

```
>>> import nltk
>>> import re
>>> nltk.data.show_cfg('grammars/book_grammars/feat0.fcfg')
```

Otro ejemplo, completamente original:

```
>>> Sentence01 = 'Kim likes children'.split()
>>> from nltk import load_parser
>>> cp = load_parser('grammars/book_grammars/feat0.fcfg', trace=2)
>>> trees = cp.nbest_parse(Sentence01)
```

Gramáticas léxicas en NLTK (1)

Esta gramática también me permite generar estructuras arbóreas:

```
>>> for tree in trees: print tree
```

Y puedo añadir tantos rasgos como crea necesario, p.e., los rasgos de tiempo, numero y concordancia en verbos:

```
print nltk.FeatStruct("[POS='N', AGR=[PER=3, NUM='pl', GND='fem']]")
```

Nuestra gramática puede llegar a ser tan compleja como queramos, p. e.:

```
nltk.data.show_cfg('grammars/book_grammars/feat1.fcfg')
```

Gramáticas léxicas en NLTK (2)

Otro ejemplo:

```
>>> Sentence02 = 'who do you claim that you like'.split()

>>> cp02 = load_parser('grammars/book_grammars/feat1.fcfg')

>>> for tree in cp02.nbest_parse(Sentence02):
        print tree

>>> tree.draw()
```

Gramáticas léxicas en NLTK (3)

Todo se ve bien, pero hay un gran problema en NLTK: antes de correr nuestro parser, necesitamos un corpus con ítems que ya tengan asignados previamente estos rasgos.

Los únicos que cumplen con estos rasgos están en inglés y alemán. Si retomamos el problema de marcación de casos, con NLTK podemos resolverlo. P. e.:

a.	Die	Katze	sieht	den	Hund
	the.NOM.FEM.SG	cat.3.FEM.SG	see.3.SG	the.ACC.MASC.SG	dog.3.MASC.SG
	'the cat sees the dog'				
b.	*Die	Katze	sieht	dem	Hund
	the.NOM.FEM.SG	cat.3.FEM.SG	see.3.SG	the.DAT.MASC.SG	dog.3.MASC.SG
c.	Die	Katze	hilft	dem	Hund
	the.NOM.FEM.SG	cat.3.FEM.SG	help.3.SG	the.DAT.MASC.SG	dog.3.MASC.SG
	'the cat helps the dog'				
d.	*Die	Katze	hilft	den	Hund
	the.NOM.FEM.SG	cat.3.FEM.SG	help.3.SG	the.ACC.MASC.SG	dog.3.MASC.SG

Gramáticas léxicas en NLTK (4)

```
>>> nltk.data.show_cfg('grammars/book_grammars/german.fcfg')
```

Dentro de lo posible, puedo analizar nuevas oraciones conforme a esta gramática de rasgos para el alemán, p.e.: *Ich folge den Katzen* (*"yo sigo a* los gatos").

```
>>> Sentence03 = 'ich folge den Katzen'.split()

>>> cp03 = load_parser('grammars/book_grammars/german.fcfg')

>>> for tree in cp03.nbest_parse(Sentence03):
...     print tree

>>> tree.draw()
```

Gracias por su atención

Blog del curso:
http://cesaraguilar.weebly.com/introduccioacuten-a-la-linguumliacutestica-computacional.html