Apuntes de Gramáticas lógicas

Javier Béjar - Antonio Jerez

Departament de Llenguatges i Sistemes Informàtics

FIB Facultat d'Informàtica de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

1. Gramáticas lógicas

1.1 Análisis como deducción

Las gramáticas lógicas son un formalismo para el tratamiento del lenguaje natural que se basa en la suposición de que podemos identificar el análisis del lenguaje natural con una demostración lógica.

De esta manera podemos describir una gramática como si se tratara de un razonamiento en el que el objetivo es demostrar si una frase es correcta.

En este razonamiento tenemos dos elementos, en primer lugar, la frase que queremos analizar, que podríamos considerar como los hechos de nuestro razonamiento. En segundo lugar, tenemos la gramática que nos dice que frases consideramos correctas y el lexicón que nos indica qué palabras conocemos y sus categorías gramaticales, estos dos elementos serían las premisas de nuestro razonamiento que nos permitirán hacer las deducciones que nos llevaran a demostrar la validez de una frase.

Si hiciéramos analogía con los sistemas de producción, la frase sería los hechos de nuestro problema y la gramática y el lexicón las reglas de nuestra base de conocimiento.

Puestos en este escenario, el analizar una frase se transforma en una demostración, en la que sabremos si una frase es correcta si el razonamiento que planteamos lo es.

Ejemplo 1 *Supongamos que queremos analizar la frase "El clima es caluroso". Para poder analizar una frase de este estilo necesitamos una gramática que describa su estructura como la siguiente:*

```
snom, sverb -> f
determinante, nombre -> snom
verbo, compl -> sverb
adjetivo -> compl
```

Para poder transformar el análisis en un proceso de razonamiento, supongamos que queremos demostrar que entre diferentes posiciones de una frase aparecen los elementos gramaticales que queremos reconocer, de manera que introducimos en nuestra gramática las posiciones entre las que reconocemos cada elemento

```
snom(X,Z), sverb(Z,Y) -> f(X,Y)
determinante(X,Z), nombre(Z,Y) -> snom(X,Y)
verbo(X,Z), compl(Z,Y) -> sverb(X,Y)
adjetivo(X,Y) -> compl(X,Y)
```

Para poder demostrar para una frase concreta si es correcta o no bastaría indicar las posiciones en las que aparece cada palabra

```
el(1,2)
clima(2,3)
es(3,4)
caluroso(4,5)
```

E introducir un lexicón que vincule cada palabra con su categoría gramatical

```
el(X,Y) -> determinante(X,Y)
clima(X,Y) -> nombre(X,Y)
es(X,Y) -> verbo(X,Y)
caluroso(X,Y) -> adjetivo(X,Y)
```

Con todos estos elementos ya tenemos el razonamiento que queremos demostrar, deseamos saber si hay una frase correcta entre las posiciones 1 y 5.

```
f(1,5)
```

1.2 El lenguaje Prolog

Una posibilidad de implementar las gramáticas lógicas es usar algún lenguaje que nos permita plantear y demostrar razonamientos lógicos. El mas extendido es el lenguaje PROLOG[1].

Resumiendo las características del lenguaje PROLOG, lo podemos describir como un lenguaje de tipo declarativo, es decir, no indicamos los pasos a seguir para resolver un problema, sino que declaramos las propiedades que debe cumplir la solución. Esto implica que es el propio lenguaje el que tienen que buscar los pasos para obtener la solución que cumple esas propiedades.

El fundamento de este lenguaje es la lógica de primer orden, de manera que un programa constará de un conjunto de declaraciones descritas en esta lógica. Dado que el coste de una demostración cualquiera puede tener un coste prohibitivo, se utiliza sólo un subconjunto de las expresiones que se pueden construir en lógica de primer orden[2].

Dado que el lenguaje ha de buscar la solución que cumple las propiedades que declaramos, este necesita un mecanismo de razonamiento. Este se basa en el razonamiento hacia atrás, que a efectos prácticos es una búsqueda en profundidad con backtracking[3].

1.2.1 La sintaxis de PROLOG

Al ser un lenguaje basado en lógica de primer orden la sintaxis es una traducción de los elementos de la lógica a un lenguaje de programación. Sus elementos son:

Constantes: Elementos identificables del dominio, se representan mediante cadenas de caracteres en minúsculas (`pepe`, `a`)

Variables: Elementos indeterminados del dominio, se representan mediante cadenas de caracteres en mayúsculas o cadenas que empiezan por el carácter subrayado (`X`, `_var`)

Predicados: Relaciones entre elementos, se representan mediante cadenas de caracteres en minúsculas y sus parámetros se encierran entre paréntesis y se separan por comas (`padre(pepe,juan)`, `mayor(X,Y)`)

Reglas: Relaciones deductivas entre predicados. Una regla en PROLOG se escribe poniendo primero el consecuente (sólo un consecuente) y después el antecedente, separándolos mediante la conectiva implicación ($\rightarrow$), que se transforma en el símbolo `:-`. Los predicados del antecedente pueden estar combinados mediante el operador de conjunción ($\wedge$), que se escribe como una coma (`,`) y el operador de disyunción ($\vee$), que se escribe como un punto y coma (`;`)

```
abuelo(X,Y) :- padre(X,Z), padre(Z,Y).
```

[1] Podéis encontrar documentación sobre el funcionamiento del lenguaje PROLOG en la web de la asignatura.

[2] Concretamente sólo se pueden utilizar cláusulas de Horn.

[3] En realidad el mecanismo de razonamiento de PROLOG implementa el algoritmo de resolución que conocéis de ILO

Toda expresión en PROLOG acaba en un punto.

Evidentemente el lenguaje PROLOG tiene otros elementos que podemos encontrar en cualquier lenguaje de programación, como por ejemplo tipos de datos primitivos (enteros, reales, cadenas, listas), operaciones sobre estos tipos de datos, etc.

Un programa PROLOG estará formado por un conjunto de reglas y predicados que expresan las propiedades de la solución que buscamos.

Ejemplo 2 *Podemos por ejemplo definir las condiciones que cumple un número para ser el factorial del otro*

```
1 fact(1,1).
2 fact(X,Y) :- X1 is X-1, fact(X1,Y1), Y is X * Y1.
```

Este programa declara que el factorial de `1` *es* `1` *y que para que* `Y` *sea el factorial de* `X` *han de cumplirse 3 condiciones, que haya un* `X1` *que sea* `X-1`*, que exista un* `Y1` *que sea el factorial de* `X1` *y que* `Y` *sea el producto de* `X` *e* `Y1`*.*

Aunque parezca una versión del programa habitual para el calculo del factorial en cualquier lenguaje imperativo, en realidad no lo es. PROLOG nos permite hacer cosas que no podemos en imperativo.

Dado que un interprete de PROLOG ha de buscar los valores de las variables que cumplen el programa nosotros le podemos pedir sin tener que cambiar una línea que nos verifique por ejemplo:

`fact(3,6).` *(¿6 es el factorial de 3?)*
`fact(3,X).` *(¿Qué X cumple que sea el factorial de 3?)*
`fact(X,120).` *(¿Qué X cumple que tenga como factorial 120?)*
`fact(X,Y).` *(¿Qué X son factorial de Y? De hecho estamos pidiendo que se calculen todos los factoriales que existen)*

Evidente para hacer lo mismo en un progama imperativo deberiamos escribir un programa para cada pregunta.

Hay varias cosas que hay que tener presentes para entender un programa PROLOG

- La especificación de las condiciones de un problema se expresan por lo general de manera recursiva
- Cada regla de un programa es independiente de las demás, por lo que las variables que aparecen son locales
- Las variables que aparecen en los predicados **no son parámetros** como los de un lenguaje imperativo
- Los nombres de las variables no significan nada, que dos variables se llamen igual en dos reglas no quiere decir que sean la misma
- El mecanismo de ejecución de un programa se basa en el razonamiento hacia atrás, por lo que partimos de un objetivo e intentamos verificar que podemos deducirlo a partir del programa
- La ejecución de las reglas de un programa PROLOG se realiza en el orden en que aparecen, eso determina el orden de ejecución en el caso de que haya más de una regla aplicable
- El razonamiento hacia atrás implica unificación de variables entre los objetivos y los consecuentes de las reglas (nada de paso de parámetros)
- En PROLOG no existe la asignación
- La programación en un lenguaje declarativo es muy diferente a la de un lenguaje imperativo, estamos indicando las condiciones que han de cumplir las soluciones que queremos, no la manera de hallarlas.

1.3 Análisis como programa PROLOG

A partir de esta sintaxis podemos fácilmente transformar cualquier gramática en un programa PROLOG y utilizar un intérprete de este lenguaje para analizar frases

Ejemplo 3 *Si tomamos el ejemplo inicial podemos transformarlo en un programa PROLOG*

```
1  f(X,Y) :- snom(X,Z), sverb(Z,Y).
2  snom(X,Y) :- determinante(X,Z), nombre(Z,Y).
3  sverb(X,Y) :- verbo(X,Z), compl(Z,Y).
4  compl(X,Y) :- adjetivo(X,Y).
5
6  determinante(X,Y) :- el(X,Y).
7  nombre(X,Y) :- clima(X,Y).
8  verbo(X,Y) :- es(X,Y).
9  adjetivo(X,Y) :- caluroso(X,Y).
10
11 el(1,2).
12 clima(2,3).
13 es(3,4).
14 caluroso(4,5).
```

Y podríamos preguntar `f(1,5)`

Si hacemos una traza de la ejecución del proceso de demostración nos encontramos con los siguientes pasos (`O` *indica los objetivos a cumplir,* `R` *indica las reglas aplicables,* `H` *indica los hechos aplicables, se han renombrado algunas variables durante la ejecución para evitar confusiones):*

```
(O) {f(1,5)}
(R) {(f(X,Y) :- snom(X,Z), sverb(Z,Y)),X=1,Y=5}
(O) {snom(1,Z), sverb(Z,5)}
(R) {(snom(Z,Y) :- determinante(X,Z), nombre(Z,Y)), X=1,Y=Z}
(O) {determinante(1,Z1), nombre(Z1,Z),sverb(Z,5)}
(R) {(determinante(X,Y) :- el(X,Y)),X=1,Y=Z1}
(O) {el(1,Z1), nombre(Z1,Z),sverb(Z,5)}
(H) {el(1,2), Z1=2}
(O) {nombre(2,Z),sverb(Z,5)}
(R) {(nombre(X,Y) :- clima(X,Y)),X=2,Y=Z}
(O) {clima(2,Z),sverb(Z,5)}
(H) {clima(2,3),Z=3}
(O) {sverb(3,5)}
(R) {(sverb(X,Y) :- verbo(X,Z), compl(Z,Y)),X=3,Y=5}
(O) {verbo(3,Z), compl(Z,5)}
(R) {(verbo(X,Y) :- es(X,Y)),X=3,Y=Z}
(O) {es(3,Z),compl(Z,5)}
(H) {es(3,4),Z=4}
(O) {compl(4,5)}
(R) {(compl(X,Y) :- adjetivo(X,Y)),X=4,Y=5}
(O) {adjetivo(4,5)}
(R) {(adjetivo(X,Y) :- caluroso(X,Y)),X=4,Y=5}
(O) {caluroso(4,5)}
(H) {caluroso(4,5)}
(O) {}
```

Por lo tanto se puede demostrar que hay una frase correcta entre las posiciones 1 y 5.

1.4 Gramáticas de cláusulas definidas

Parece evidente que transformar directamente a la sintaxis PROLOG una gramática puede ser un poco farragoso al tener que introducir los puntos de la frase donde esta cada palabra y tener que tratar en las reglas de la gramática esos puntos. Además es complicado tener mezcladas las reglas de la gramática y el lexicón.

Para hacer más fácil las cosas PROLOG implementa lo que denominaremos *gramáticas de cláusulas definidas*. Estas permiten ocultar todos los elementos de la transformación que podríamos decir que son más cuestión de implementación.

Las gramáticas de cláusulas definidas nos permitirán escribir gramáticas incontextuales, con la ventaja de que tendremos la oportunidad de introducir código PROLOG dentro de las reglas que nos ayudará a tratar todo aquello que es necesario en el tratamiento del lenguaje natural y que no permiten las gramáticas incontextuales, y además usar variables para pasar valores de una regla de la gramática a otra.

La sintaxis de las gramáticas de las cláusulas definidas varía ligeramente respecto a la de PROLOG, cambiando el símbolo `:-` por `-->`. Se ha de distinguir lo que son elementos de la gramática (terminales y no terminales), de lo que es el código PROLOG que añadimos a las producciones poniendo ese código entre llaves.

```
1 f --> snom, sverb.
2 snom --> det(G1,N1),nom(G2,N2),{G1=G2,N1=N2}.
```

Para no tener que transformar las frases que queremos analizar, las gramáticas de cláusulas definidas nos permiten representar las frases como listas de palabras. Las palabras deberán introducirse en el lexicón como predicados PROLOG, de esta manera lo separaremos de las reglas de la gramática y tendremos una representación mas adecuada para el lexicón.

Adicionalmente disponemos de un operador que nos permite consumir las palabras de la frase de entrada. Este operador se corresponde con el operador de listas de PROLOG y se representa mediante corchetes. Cuando queremos consumir algo de la entrada debemos poner las variables o símbolos que queramos extraer de la entrada entre corchetes y separados por comas.

```
1 det(G,N) --> [W],{determinante(W,G,N)}.
2 determinante(el,masculino,singular)
```

Ejemplo 4 *Esta sería la gramática ejemplo transformada a la sintaxis de gramáticas de cláusulas definidas:*

```
1  /* Gramática */
2
3  f --> snom, sverb.
4  snom --> determinante, nombre.
5  sverb --> verbo, compl.
6  compl --> adjetivo.
7
8  determinante --> [W], {ldeterminante(W)}.
9  nombre --> [W], {lnombre(W)}.
10 verbo --> [W], {lverbo(W)}.
```

```
11 adjetivo --> [W], {ladjetivo(W)}.
12
13 /* lexicon */
14
15 ldeterminante(el).
16 lnombre(clima).
17 lverbo(es).
18 ladjetivo(caluroso).
```

Para analizar una frase le haremos la siguiente pregunta a PROLOG:
`f([el,clima,es,caluroso],[])`[4]
Podemos seguir la ejecución del análisis de la gramática, en este caso tenemos las ejecución de las reglas de la gramática y la ejecución del código PROLOG, la lista de entrada queda oculta.

```
(O) f([el,clima,es,caluroso],[])
(R) {f --> snom, sverb}
(O) {snom, sverb}
(R) {snom --> determinante, nombre}
(O) {determinante, nombre, sverb}
(R) {determinante --> [W], {ldeterminante(W)}}
 Consumimos el primer elemento de la lista W=el
(O) {ldeterminante(el), nombre, sverb}
(H) {ldeterminante(el)}
(O) {nombre, sverb}
(R) {nombre --> [W], {lnombre(W)}}
 Consumimos el siguiente elemento de la lista W=clima
(O) {lnombre(clima), sverb}
(H) {lnombre(clima)}
(O) {sverb}
(R) {sverb --> verbo, compl}
(O) {verbo, compl}
(R) {verbo --> [W], {lverbo(W)}}
 Consumimos el siguiente elemento de la lista W=es
(O) {lverbo(es), compl}
(H) {lverbo(es)}
(O) {compl}
(R) {compl --> adjetivo}
(O) {adjetivo}
(R) {adjetivo --> [W], {ladjetivo(W)}}
 Consumimos el siguiente elemento de la lista W=caluroso
(O) {ladjetivo(caluroso)}
(H) {ladjetivo(caluroso)}
(O) {}
```

Por lo tanto la frase es correcta

Hemos de tener en cuenta también lo siguiente:

- El análisis de una frase no es correcto si no se consume toda la entrada

[4]El segundo parámetro es siempre una lista vacía y es una lista auxiliar que necesitan las gramáticas de cláusulas definidas

- Si queremos generar una salida a partir del análisis hemos de modificar la gramática adecuadamente y añadir variables a la producción principal de la gramática que se unifiquen a lo que queramos generar

2. Aproximación práctica a las Gramáticas Lógicas

2.1 Estructura de constituyentes

La gramática 1 se ha realizado exclusivamente para el análisis de las frases:

(a) Juan ríe.

(b) Juan piensa en clara.

(c) Un profesor habla con Clara.

(d) Juan está en Barcelona.

(e) El hombre lee un libro

Las frases presentadas son oraciones simples y están compuestas de sujeto verbo y, en algunos casos, complementos verbales. El objetivo de este ejemplo es la construcción de una gramática donde se expresen los constituyentes de la oración. Veamos el nivel sintáctico:

```
1  aserción --> sn, verb, compl.
2  compl --> [].
3  compl --> prep, sn.
4  compl --> sn.
5  sn --> npr.
6  sn --> det, n.
7
8  verb --> [W], {verbo(W)}.
9  npr --> [W], {npropio(W)}.
10 n --> [W], {nombre(W)}.
11 det --> [W], {determ(W)}.
12 prep --> [W], {prepo(W)}.
```

Como vemos el sintagma nominal (`sn`), que realiza la función sujeto en las oraciones presentadas y es el primer componente del nodo 'aserción', presenta dos tipos de construcción:

1. los sintagmas nominales compuestos por un único elemento, nombres propios (`npr`) correspondientes a las frases (a), (b) y (d) y

2. los sintagmas nominales compuestos por mas de un elemento (determinante y nombre común, "el hombre" y "un profesor") que corresponden a las frases (c) y (e)

El segundo elemento del nodo aserción es una categoría léxica (`verb`), y el tercero es un nodo sintagmático que analiza los complementos verbales (`comp`) cuya composición describimos mediante tres reglas. En primer lugar debemos expresar la posibilidad de que este nodo no se realice, como en la frase (a), el verbo 'ríe' se utiliza en este caso de forma intransitiva y por lo tanto no tiene complemento verbal. Las siguientes opciones de `compl` analizan los sintagmas preposicionales (frases

(b), (c), y (d)), compuestos de preposición y sintagma nominal, y los sintagmas nominales para los que utilizamos la regla `sn` anteriormente descrita.

Todos los elementos terminales se declaran con una serie de reglas donde se indica el reconocimiento de una palabra en la lista de entrada a analizar (frase) y la condición que debe cumplir en el módulo léxico, que en este caso es la categorización léxica de cada una de las palabras. Así la regla "`verb->[W], {verbo(W)}`." indica que la palabra `W` debe cumplir la condición de estar categorizada como 'verbo' en el léxico. En el fichero correspondiente al módulo léxico deberemos categorizar cada uno de los elementos léxicos de la frase:

```
npropio(clara).    npropio(maria).   npropio(juan).   nombre(hombre).
nombre(profesor).  nombre (libro).   determ(el).      determ(un).
verbo(esta).       verbo (rie).      verbo(piensa).   verbo(habla).
verbo(lee).        prepo(en).        prepo(con).      prepo(de).
```

Esta gramática es capaz de analizar las frases presentadas y muchas mas, incluso algunas que se consideran agramaticales. Por ejemplo la frase (f)* "Juan piensa de María" o (g) * "María habla en Juan", son frases incorrectas.

2.1.1 Restricciones de selección argumental

El objetivo de la gramática 2 será conseguir que las reglas de la gramática lleven a cabo la selección del tipo de complemento de cada verbo y, por lo tanto, no acepte frases como (f) y (g). Para ello debemos introducir la información sobre el tipo de preposición que acepta cada verbo. Veamos como describiremos esta exigencia en la gramática.

```
1  asercion --> npr, verb(X), compl(X).
2  compl([]) --> [].
3  compl([arg(X)]) --> prep(X),sn.
4  compl([]) --> sn.
5  sn --> npr.
6  sn --> det, n.
7
8  verb(A) --> [W], {verbo(W,A)}.
9  npr --> [W], {npropio(W)}.
10 n --> [W], {nombre(W)}.
11 det --> (W], {determ{W)}.
12 prep(W) --> [W], {prepo(W)}.
```

El complemento (`compl`) toma como argumento el valor de la preposición de la que está formado. Este valor se etiqueta como `arg` y se incluye en una lista de posibles valores que deberá unificar con la lista de valores declarados en la entrada léxica verbal. A su vez `prep` propaga el valor de la preposición hacia el complemento. Por lo tanto, es necesario realizar algunos cambios a nivel léxico, para cada verbo se deben indicar los argumentos preposicionales que acepta. El resto de entradas léxicas no necesitan modificación.

```
verbo(piensa,[arg(en)]).
verbo(esta, [arg(en)]).
verbo(rie, []).
verbo(habla, [arg(con)]).
verbo (lee, []).
```

El ultimo paso es la declaración a nivel de frase de los valores que deben unificar.

```
asercion --> sn, verb(X),comp(X).
```

La gramática 2 no acepta frases como: (*) Juan piensa con María (y naturalmente otras muchas correctas pero no contempladas) que si aceptaba la gramática 1.

Hasta el momento hemos tratado las restricciones de selección de complemento para verbos sin complemento verbal (intransitivos) y para verbos de un argumento. Tomemos como siguiente objetivo la posibilidad de que un verbo acepte más de un complemento, ampliaremos el corpus inicial con la frase (h) "Juan habla de Clara con María".

La posibilidad de que un verbo rija más de un complemento hace que debamos trabajar con listas de posibles preposiciones regidas por el verbo, es decir listas con más de un elemento. Es necesario por lo tanto ampliar las reglas de complemento:

```
compl([arg(X) | Y]) --> prep(X), sn, compl(Y).
```

También es necesaria la declaración a nivel léxico de la lista de argumentos del verbo. Veamos ahora la definición de 'habla' en el léxico:

```
verbo(habla, [arg(de),arg(con)]).
```

Gramática 1

Esta es la primera gramática completa.

```
analisis(X,Y):- asercion(X,Y).

asercion --> sn, verb,compl.
compl --> [].
compl --> prep,sn.
compl--> sn.
sn--> npr.
sn--> det,n.

verb--> [W],{verbo(W)}.
npr--> [W],{npropio(W)}.
n--> [W],{nombre(W)}.
det--> [W],{determ(W)}.
prep--> [W],{prepo(W)}.

npropio(clara).
npropio(maria).
npropio(juan).
npropio(barcelona).
nombre(hombre).
nombre(profesor).
nombre(libro).
determ(un).
determ(el).
determ(un).
verbo(esta).
```

```
verbo(rie).
verbo(piensa).
verbo(habla).
verbo(lee).
prepo(en).
prepo(con).
prepo(de).
```

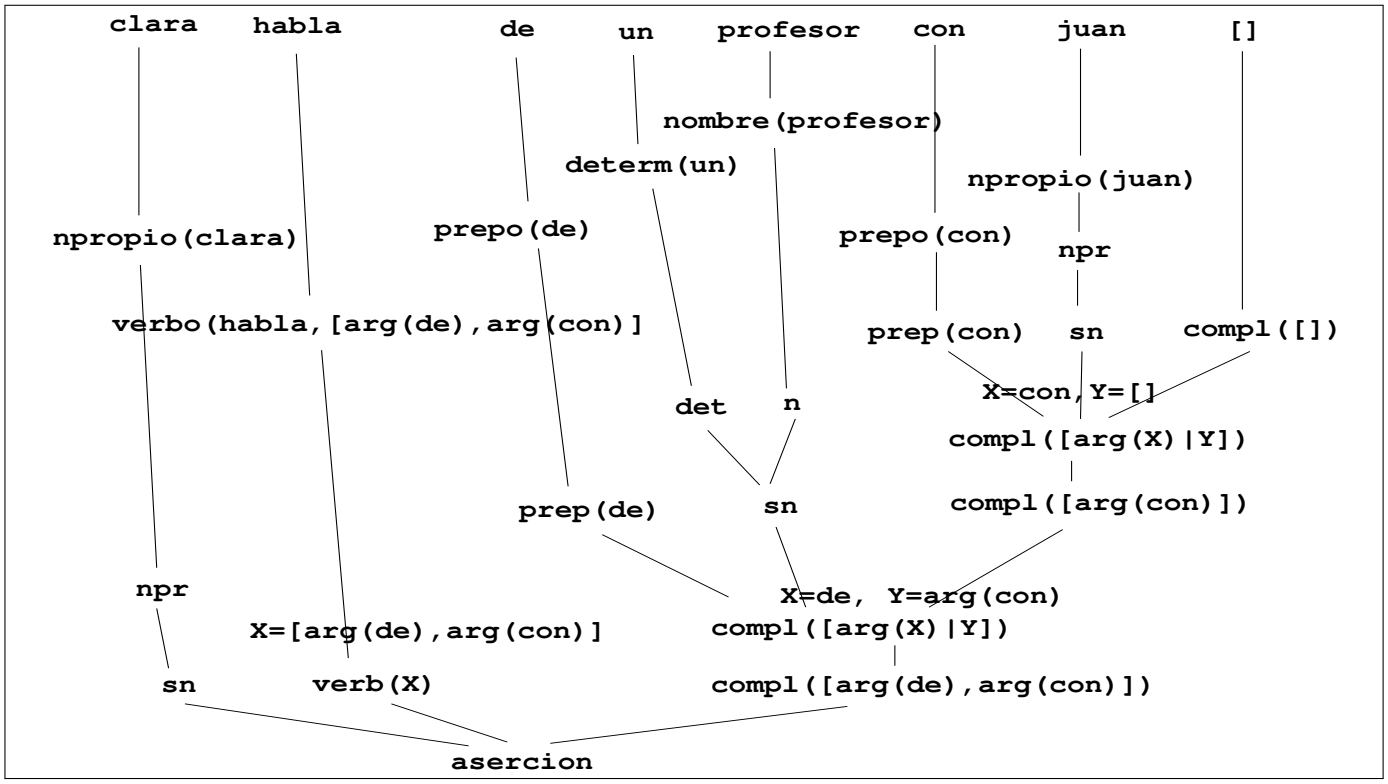
Gramática 2

Esta es la segunda gramática completa.

```
analisis(X,Y):- asercion(X,Y).

asercion --> sn, verb(X),compl(X).

compl([])--> [].
compl([])--> sn.
compl([arg(X)|Y])--> prep(X),sn,compl(Y).

sn-->npr.
sn-->det,n.

verb(A)--> [W],{verbo(W,A)}.
npr--> [W],{npropio(W)}.
n--> [W],{nombre(W)}.
det--> [W],{determ(W)}.
prep(W)--> [W],{prepo(W)}.

npropio(clara).
npropio(maria).
npropio(juan).
npropio(barcelona).

nombre(hombre).
nombre(profesor).
nombre(libro).

determ(un).
determ(el).

prepo(en).
prepo(con).
prepo(de).

verbo(piensa,[arg(en)]).
verbo(esta,[arg(en)]).
verbo(rie,[]).
verbo(habla,[arg(de),arg(con)]).
verbo(lee,[]).
```

2.1.2 Ejemplo de análisis

Este es un ejemplo del analisis de la frase "Clara habla de un profesor con Juan" utilizando la gramatica 2.



2.1.3 Ejercicios

1. A partir de la gramática 1, introducir información sobre género y número a nivel léxico y expresar las condiciones de buena formación de las frases. La concordancia se puede establecer a nivel de sintagma nominal (el determinante y el nombre deben concordar en género y número) y a nivel de aserción (el 'sn' sujeto y el 'sv' deben concordar en número).

2. A partir de la gramática 1, definir las reglas necesarias para que ésta analice la frase:

   (i) El hombre que habla de maria rie.

   En esta frase, el sintagma nominal está formado por un determinante , un nombre y una oración subordinada de relativo.

3. Nótese que la gramática 2 es capaz de analizar la frase: (h) "Juan habla de Clara con María" pero no la frase (j) "Juan habla con Maria de Clara". Añadir algunas reglas Prolog que permitan que el orden de aparición de los complementos no esté determinado.

2.2 Representación lógica

El segundo objetivo que nos hemos trazado es la construcción de una representación lógica de la frase. La formula lógica se construye mediante la aplicación de las reglas de forma compositiva y la representación que queremos obtener se debe declarar en el nivel léxico.

En la gramática 3 la representación que obtendremos será considerar el verbo (núcleo de la frase) como un predicado lógico con argumentos, que corresponderán al sujeto y complementos verbales. Por ejemplo:

Juan piensa en Maria = pensar(juan, maria).

La aridad del predicado lógico variará, por lo tanto, en función del verbo que represente. Por ejemplo:

predicado unario `verbo(reir, S,[],reir(S)).`

predicado binario `verbo(piensa,S,(arg(en,O)],pensar_en(S,O)).`

predicado ternario `verbo(habla,S,[arg(con,O),arg(de,O1)],comunica(S,O,O1)).`

En el caso en que los argumentos son nombres propios, éstos serán siempre constantes y nunca predicados más complejos. Así la constante designada por el nombre "clara" es '`clara`'.

```
npr(clara).
npr(maria).
npr(juan).
npr(Barcelona).
```

Los nombres comunes se representarán como predicados unarios del tipo:

`hombre(X).`[1]

Así a nivel léxico los nombres comunes se representarán del siguiente modo:

```
nombre(libro, K,libro(K)).
nombre(hombre, K,hombre(K)).
nombre(profesor,K,profesor(K)).
```

Siendo `K` la variable sobre la que se predica la propiedad de ser 'libro', 'hombre' o 'profesor'. Las unidades léxicas categorizadas como verbos son las que aportan la información sobre como va a ser la estructura resultante del análisis de la oración, hemos incluido a la lista de argumentos el valor del complemento (`O`) que, en el caso de que exista y junto con el sujeto, es uno de los argumentos del predicado.

```
verbo(rie, S,[],reir(S)).
verbo(piensa,S,[arg(en,O)],pensar_en(S,O)).
verbo(habla,S,[arg(con,O),arg(de,O1)],comunica(S,O,O1)).
verbo(esta,S, [arg(en,O)], locativo(S,O)).
verbo(lee,S, [arg (nulo,O)], leer(S, O)).
```

El resto de categorías léxicas se declaran, por el momento del mismo modo que en las gramáticas anteriores.

```
determ(el).
determ (un).
prepo(en).
prepo(con).
prepo(de).
```

Para conseguir que las representaciones declaradas a nivel léxico se propaguen hacia el nivel sintáctico, deberemos añadir los argumentos incluidos a las siguientes reglas:

[1] Nótese que ésta es una fórmula no cerrada, la cuantificación se tratará en la siguiente gramática.

```
verb(S,A,F)--> [W], {verbo(W,S,A,F)}.
npr(W)--> [W], {npropio(W)}.
n(F)--> [W], {nombre(W,K,F)}.
det--> [W], {determ(W)}.
prep(X)--> [W], {prepo(W)}.
```

La fórmula lógica se irá componiendo en el momento en que los constituyentes de la frase entren en contacto en el análisis.

```
asercion --> sn(S), verb(S,X,F), compl(X).
compl([]) --> [].
compl([arg(X,O) | Y]) --> prep(X), sn(O),compl(Y).
compl((arg(nulo,O) | Y]) --> sn(O), compl(Y).
sn(S) --> npr(S).
sn(S) --> det, n(S).
```

Si se quiere obtener la representación, ésta se deberá añadir al nivel de aserción:

```
analisis(F,X,[]):- asercion(F,X,[]).
asercion(F) --> sn(S), verb(S,X,F), compl(X).
```

En el momento de la ejecución, esta gramática nos proporciona una representación de cada una de las frases analizadas, dado el siguiente objetivo (nótese que añadimos un argumento más que se debe calcular):

```
?- analisis (F, [maria, habla, con, juan, de, clara],[]).
```

la respuesta es:

```
F = comunica(maria, juan, clara)
```

2.3 Tratamiento de la cuantificación

Consideremos ahora las frases en que participan nombres comunes, es decir no propios. Ya hemos dicho que no podemos considerarlas fórmulas bien formadas si no incluimos algún tipo de cuantificación de las variables que intervienen. Por lo tanto el uso de nombres comunes obliga a la introducción de determinantes y desde el punto de vista lógico de cuantificadores y, por supuesto, de la explicitación del alcance de éstos en la frase. Este es el objetivo de la gramática 4.

Supongamos que representamos el sintagma nominal del siguiente modo:

el libro = e(X,libro(X)).

parafraseado:

existe X y X es un libro.

Así decimos que el determinante definido singular "el" corresponde al cuantificador existencial. De un modo parecido podemos representar el sintagma nominal "Todo libro" como `a(X, libro(X))`, es decir que representamos "todo" mediante el cuantificador universal. El determinante deberá en todos los casos mostrar en su fórmula el alcance de la cuantificación. En el caso de que el determinante formara parte del sintagma nominal sujeto, este deberá incluir la fórmula verbal puesto que ésta afecta al elemento cuantificado. Así optamos por la siguiente representación, el cuantificador

existencial introduce la conectiva lógica 'and' para relacionar los predicados y el cuantificador universal la implicación 'implies'. Veamos algunas de las frases asociadas a sus representaciones lógicas correspondientes:

el libro cae = e(X, and(libro(X), cae(X)))

Juan piensa en el libro = e(X, and(libro(X), piensa(juan, X)))

todo hombre piensa en el libro = a(X, implies(hombre(X), e(Y, and(libro(Y), piensa(X,Y)))

Veamos ahora como representar, en el nivel léxico, esta traducción de la representación sintáctica a la representación lógica. Las modificaciones afectan a las reglas del determinante.

```
determ(el, K,S1,S2, e(K,and(S1,S2))).
determ(todo, K,S1,S2, a(K,implies(S1,S2))).
```

La representación del nombre se incorpora a la del determinante en la regla de sintagma nominal.

```
1 sn(K,S2,F) --> det(K,S1,S2,F),n(K,S1).
```

También debemos contemplar el caso del nombre propio que no modifica su fórmula lógica, no obstante el sintagma nominal que lo contiene debe presentar los mismos argumentos que el formado por determinante y nombre común.

```
1 sn(K,F,F) --> npr(K).
```

También debemos tener en cuenta y modificar la regla de complemento.

```
1 compl([], S,S) --> [].
2 compl([arg(X,K) | Y],S1,S) --> prep(X), sn(K,S2,S),compl(Y,S1,S2).
```

Y por ultimo la combinación y construcción de la formula global se realiza en el nodo aserción.

```
1 asercion(S)--> sn(K,S2,S), verb(K,X,S1),compl(X,S1,S2).
```

Veamos ahora el análisis de la frase "el hombre ríe":

```
?- analisis (F, [el, hombre , rie],[]).
F =e(_704, and(hombre(_704), rie(_704)))
```

Gramática 3

Esta es la tercera gramática completa.

```
1  analisis(F,X,Y):- asercion(F,X,Y).
2
3  asercion(F) --> sn(S), verb(S,X,F),compl(X).
4
5  compl([])--> [].
6  compl([arg(X,O)|Y])--> prep(X),sn(O),compl(Y).
7  compl([arg(nulo,O)|Y])--> sn(O),compl(Y).
8
9  sn(S)--> npr(S).
10 sn(S)--> det,n(S).
```

```
11
12 verb(S,A,F)--> [W],{verbo(W,S,A,F)}.
13 npr(W)--> [W],{npropio(W)}.
14 n(F)--> [W],{nombre(W,_,F)}.
15 det--> [W],{determ(W)}.
16 prep(W)--> [W],{prepo(W)}.
17
18 npropio(clara).
19 npropio(maria).
20 npropio(juan).
21 npropio(barcelona).
22
23 nombre(libro,K,libro(K)).
24 nombre(hombre,K,hombre(K)).
25 nombre(profesor,K,profesor(K)).
26
27 determ(un).
28 determ(el).
29
30 prepo(en).
31 prepo(con).
32 prepo(de).
33
34 verbo(rie,S,[],reir(S)).
35 verbo(piensa,S,[arg(en,O)],pensar_en(S,O)).
36 verbo(habla,S,[arg(de,O),arg(con,O1)],comunica(S,O,O1)).
37 verbo(habla,S,[arg(con,O),arg(de,O1)],comunica(S,O1,O)).
38 verbo(esta,S,[arg(en,O)],locativo(S,O)).
39 verbo(lee,S,[arg(nulo,O)],leer(S,O)).
```

Ejemplos de frases

```
| ?-  analisis(F,[juan,esta,en,barcelona],[]).
F = locativo(juan,barcelona) ?
yes
| ?- analisis(F,[juan,piensa,en,maria],[]).
F = pensar_en(juan,maria) ?
yes
| ?- analisis(F,[el,libro,esta,en,barcelona],[]).
F = locativo(libro(_A),barcelona) ?
yes
| ?- analisis(F,[juan,lee,un,libro],[]).
F = leer(juan,libro(_A)) ?
yes
| ?- analisis(F,[el,hombre,habla,de,juan,con,maria],[]).
F = comunica(hombre(_A),juan,maria) ?
yes
| ?- analisis(F,[el,hombre,rie],[]).
F = reir(hombre(_A)) ?
yes
```

```
| ?- analisis(F,[el,profesor,piensa,en,un,libro],[]).
F = pensar_en(profesor(_B),libro(_A)) ?
yes
```

Gramática 4

Esta es la cuarta gramática completa.

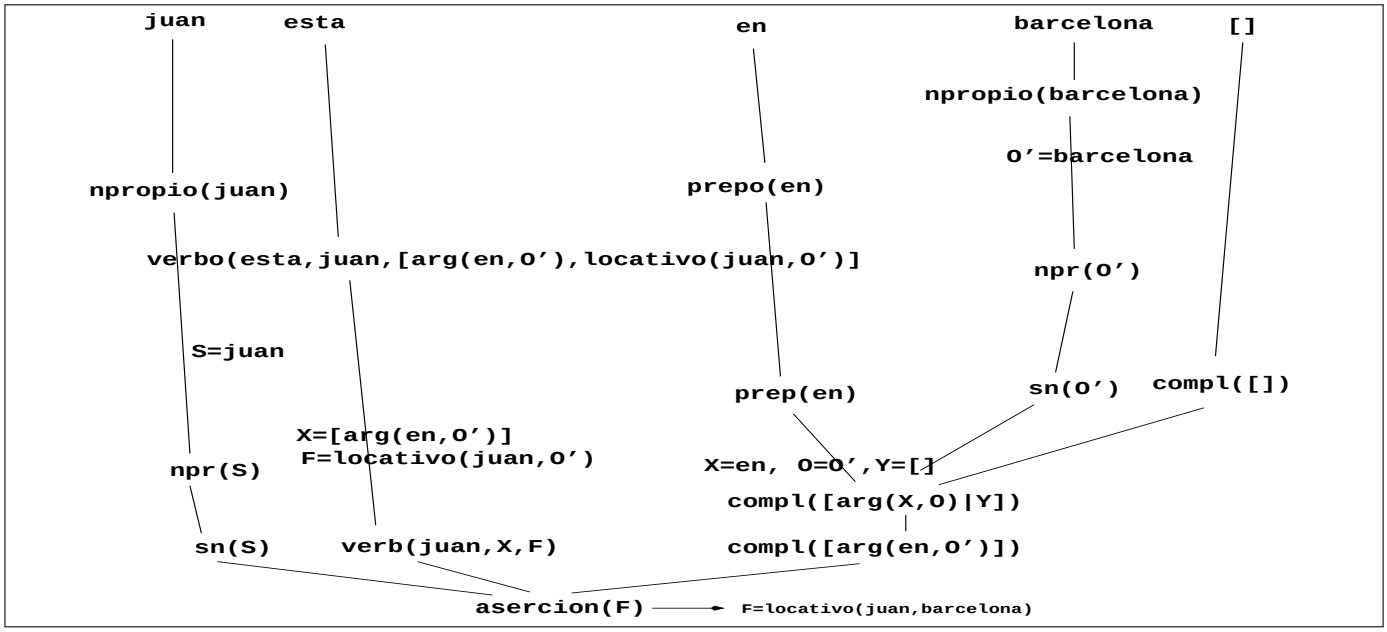
```
analisis(F,X,Y):- asercion(F,X,Y).

asercion(F) --> sn(K,S2,F), verb(K,X,S1), compl(X,S1,S2).

compl([],S,S)--> [].
compl([arg(X,K)|Y],S1,S)--> prep(X), sn(K,S2,S), compl(Y,S1,S2).
compl([arg(nulo,K)|Y],S1,S)--> sn(K,S2,S), compl(Y,S1,S2).

sn(K,F,F)--> npr(K).
sn(K,S2,F)--> det(K,S1,S2,F),n(K,S1).

verb(S,A,F)--> [W],{verbo(W,S,A,F)}.
npr(W)--> [W],{npropio(W)}.
n(K,F)--> [W],{nombre(W,K,F)}.
det(K,S1,S2,F)--> [W],{determ(W,K,S1,S2,F)}.
prep(W)--> [W],{prepo(W)}.

npropio(clara).
npropio(maria).
npropio(juan).
npropio(barcelona).

nombre(libro,K,libro(K)).
nombre(hombre,K,hombre(K)).
nombre(profesor,K,profesor(K)).

determ(el,K,S1,S2,e(K,and(S1,S2))).
determ(un,K,S1,S2,e(K,and(S1,S2))).
determ(los,K,S1,S2,a(K,implies(S1,S2))).
determ(todo,K,S1,S2,a(K,implies(S1,S2))).

prepo(en).
prepo(con).
prepo(de).

verbo(rie,S,[],reir(S)).
verbo(piensa,S,[arg(en,O)],pensar_en(S,O)).
verbo(habla,S,[arg(de,O),arg(con,O1)],comunica(S,O,O1)).
verbo(habla,S,[arg(con,O),arg(de,O1)],comunica(S,O1,O)).
verbo(esta,S,[arg(en,O)],locativo(S,O)).
verbo(lee,S,[arg(nulo,O)],leer(S,O)).
```

Ejemplos de frases

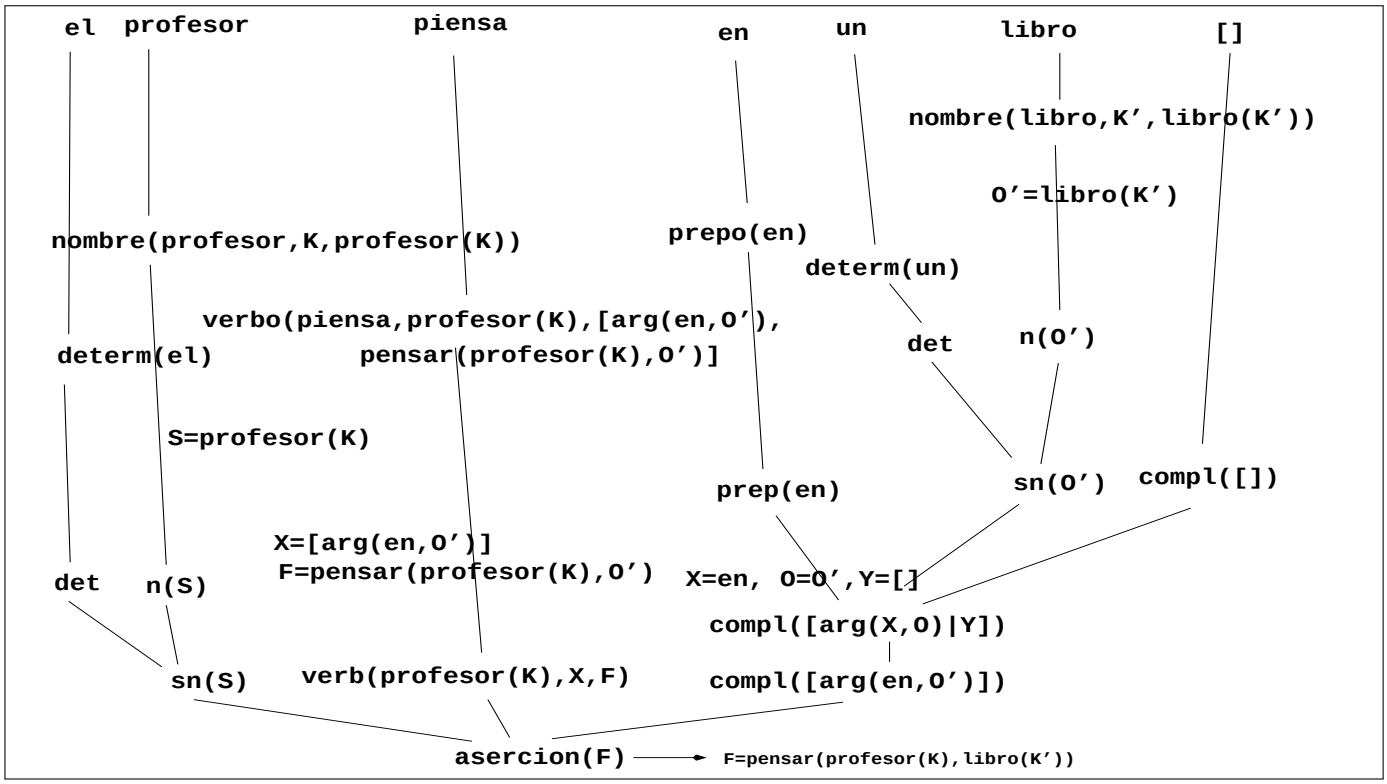
```
| ?- analisis(F,[el,hombre,rie],[]).
F = e(_A,and(hombre(_A),reir(_A))) ?
yes

| ?- analisis(F,[el,profesor,piensa,en,un,libro],[]).
F = e(_B,and(profesor(_B),e(_A,and(libro(_A),pensar_en(_B,_A))))) ?
yes

| ?- analisis(F,[el,hombre,habla,de,juan,con,maria],[]).
F = e(_A,and(hombre(_A),comunica(_A,juan,maria))) ?
yes

| ?- analisis(F,[todo,hombre,piensa,en  ,un,libro],[]).
F = a(_B,implies(hombre(_B),e(_A,and(libro(_A),pensar_en(_B,_A))))) ?
yes

| ?- analisis(F,[todo,libro,esta,en,barcelona],[]).
F = a(_A,implies(libro(_A),locativo(_A,barcelona))) ?
yes

| ?- analisis(F,[todo,libro,habla,de,un,hombre],[]).
no

| ?- analisis(F,[todo,libro,habla,de,un,hombre,con,un,profesor],[]).
F = a(_C,implies(libro(_C),e(_B,and(hombre(_B),
        e(_A,and(profesor(_A),comunica(_C,_B,_A)))))))) ?
yes
```

2.3.1 Ejemplos de análisis

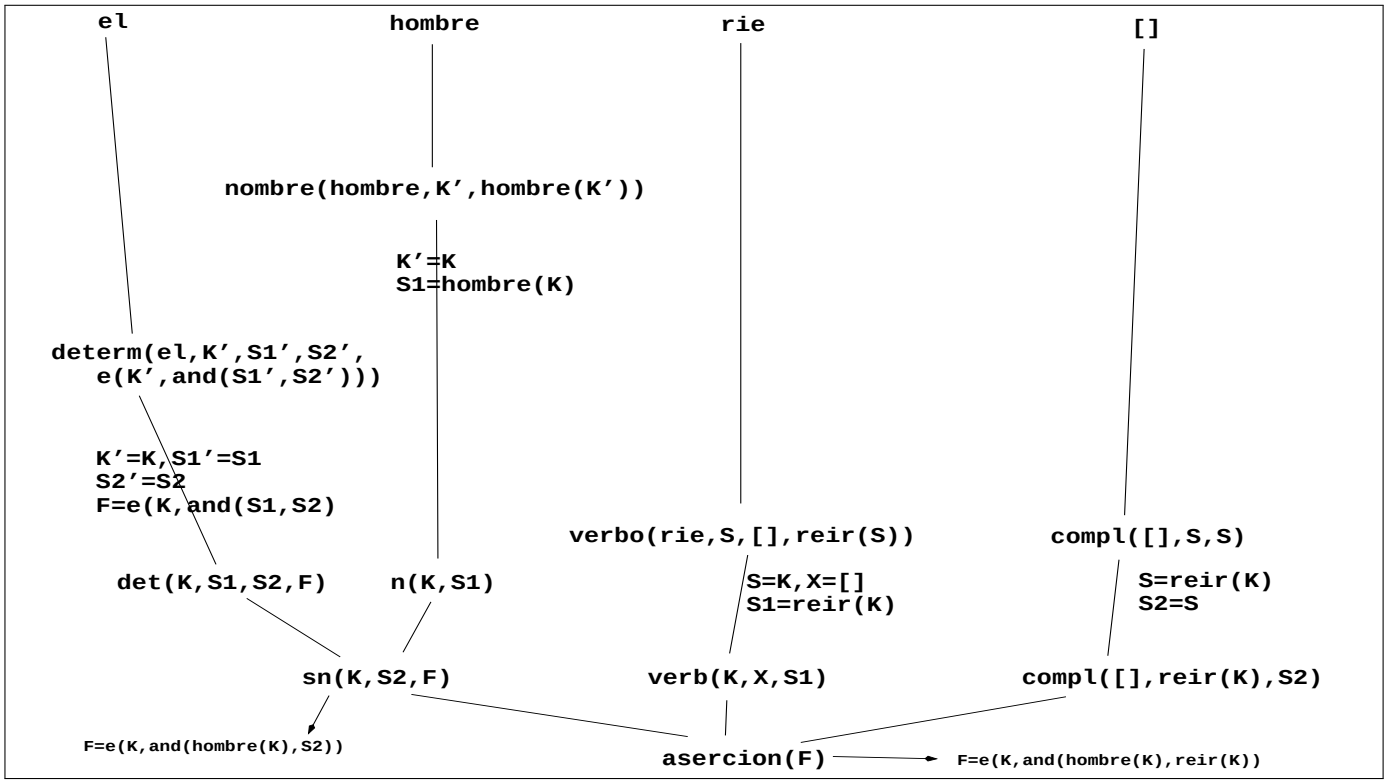
Este es un ejemplo del análisis de la frase “Juan esta en Barcelona” usando la gramática 3.



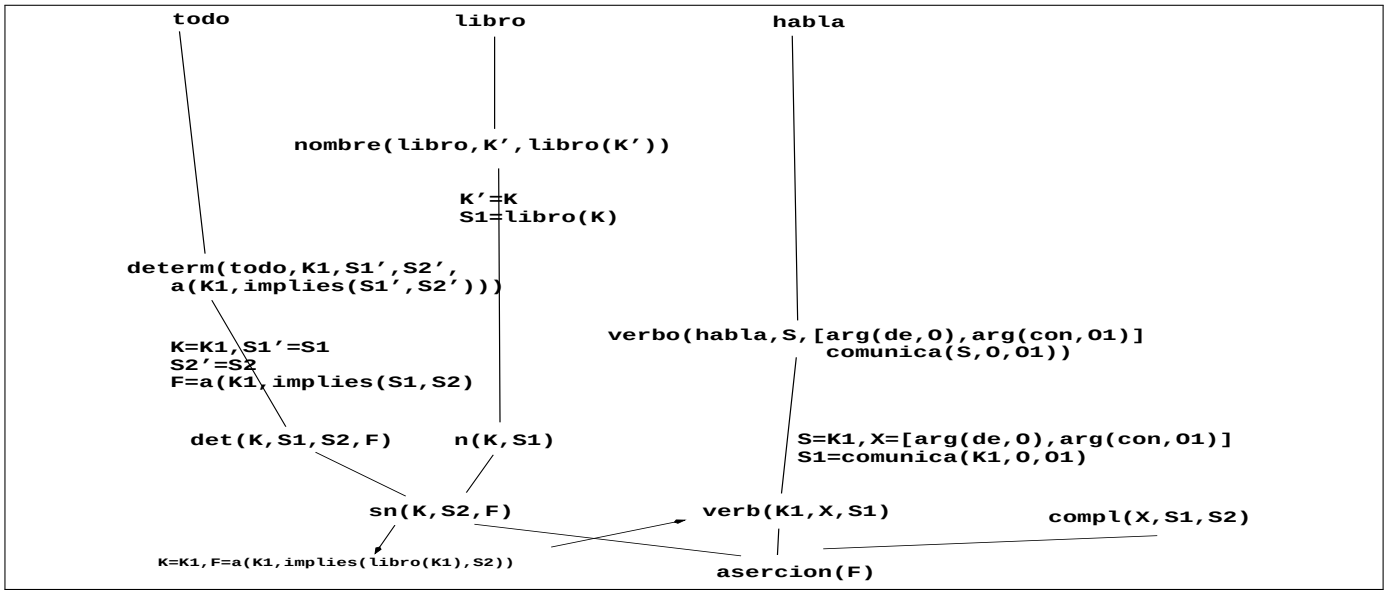
Este es un ejemplo del análisis de la frase “El profesor piensa en un libro” usando la gramática 3.

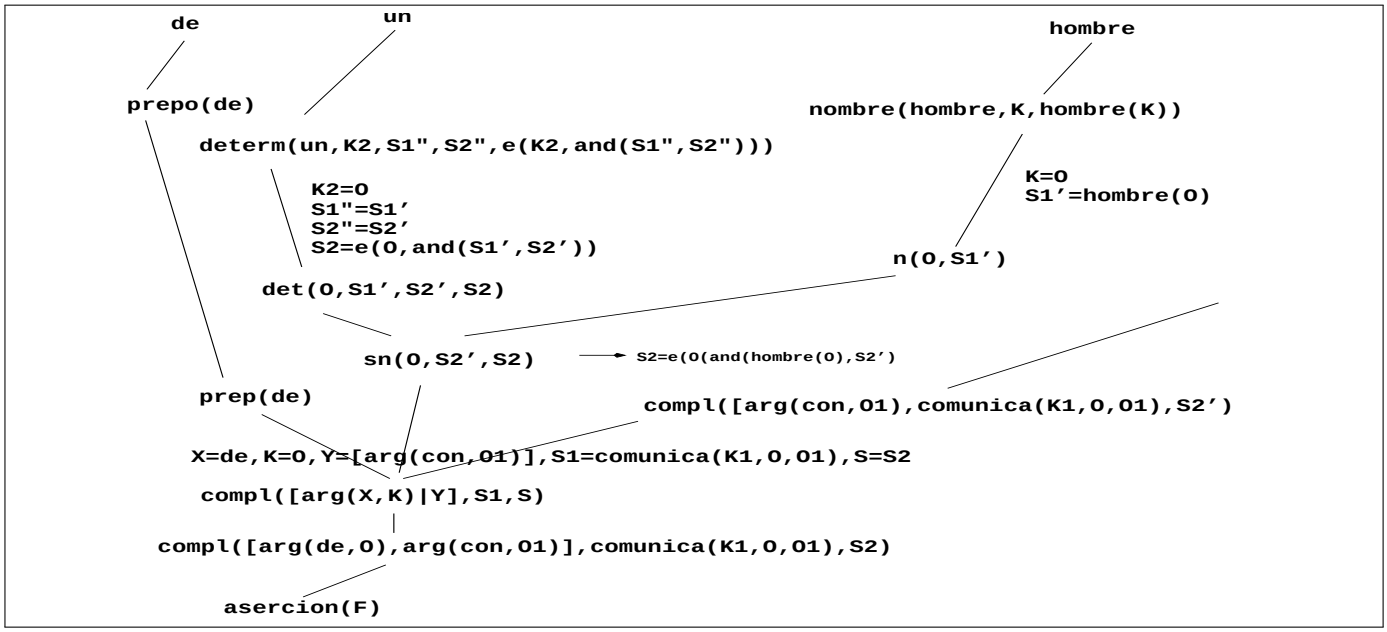


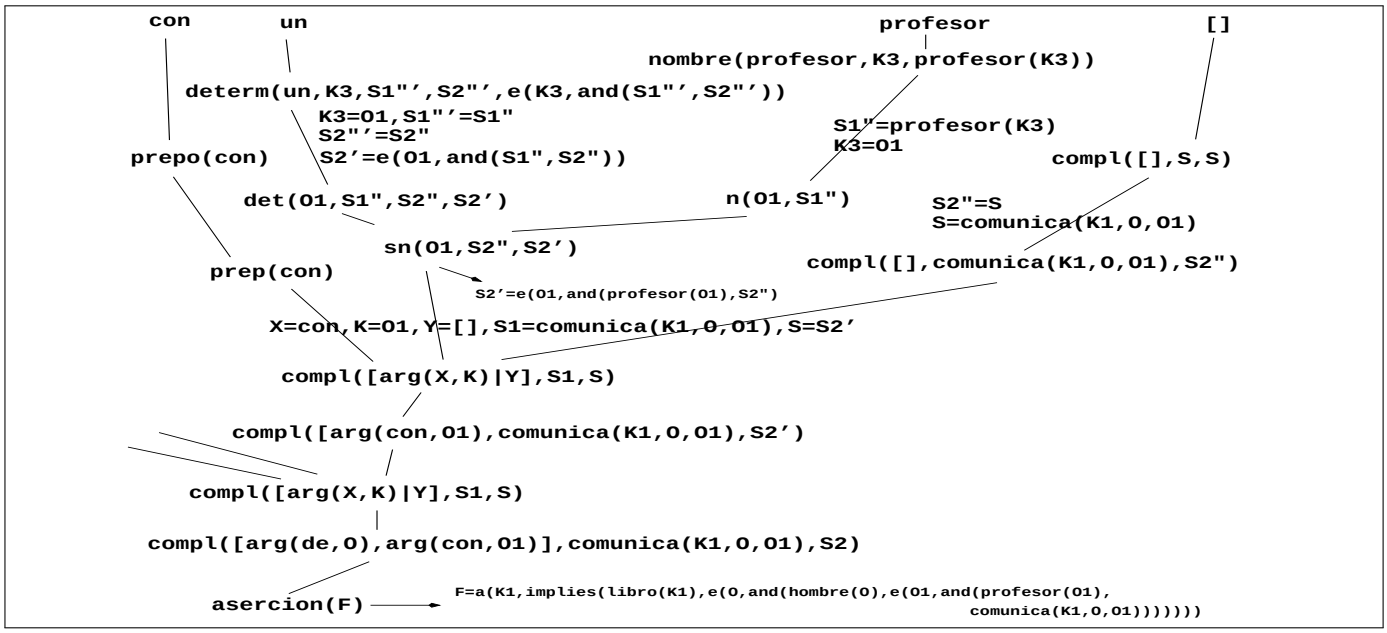
Este es un ejemplo del análisis de la frase “El hombre rie” usando la gramática 4



Este es un ejemplo del análisis de la frase “Todo libro habla de un hombre con un profesor” usando la gramática 4


todo
libro
habla
nombre(libro,K',libro(K'))
K'=K
S1=libro(K)
determ(todo,K1,S1',S2',
a(K1,implies(S1',S2')))
verbo(habla,S,[arg(de,O),arg(con,O1)]
comunica(S,O,O1))
K=K1,S1'=S1
S2'=S2
F=a(K1,implies(S1,S2)
det(K,S1,S2,F)
n(K,S1)
S=K1,X=[arg(de,O),arg(con,O1)]
S1=comunica(K1,O,O1)
sn(K,S2,F)
verb(K1,X,S1)
compl(X,S1,S2)
K=K1,F=a(K1,implies(libro(K1),S2))
asercion(F)


de
un
hombre
prepo(de)
nombre(hombre,K,hombre(K))
determ(un,K2,S1",S2",e(K2,and(S1",S2")))
K=O
S1'=hombre(O)
K2=O
S1"=S1'
S2"=S2'
S2=e(O,and(S1',S2'))
n(O,S1')
det(O,S1',S2',S2)
sn(O,S2',S2)
S2=e(O(and(hombre(O),S2')
prep(de)
compl([arg(con,O1),comunica(K1,O,O1),S2')
X=de,K=O,Y=[arg(con,O1)],S1=comunica(K1,O,O1),S=S2
compl([arg(X,K)|Y],S1,S)
compl([arg(de,O),arg(con,O1)],comunica(K1,O,O1),S2)
asercion(F)


con
un
profesor
[]
nombre(profesor,K3,profesor(K3))
determ(un,K3,S1"',S2"',e(K3,and(S1"',S2"')))
K3=O1,S1"'=S1"
S2"'=S2"
S1"=profesor(K3)
K3=O1
prepo(con)
S2'=e(O1,and(S1",S2"))
compl([],S,S)
det(O1,S1",S2",S2')
n(O1,S1")
S2"=S
S=comunica(K1,O,O1)
sn(O1,S2",S2')
compl([],comunica(K1,O,O1),S2")
prep(con)
S2'=e(O1,and(profesor(O1),S2")
X=con,K=O1,Y=[],S1=comunica(K1,O,O1),S=S2'
compl([arg(X,K)|Y],S1,S)
compl([arg(con,O1),comunica(K1,O,O1),S2')
compl([arg(X,K)|Y],S1,S)
compl([arg(de,O),arg(con,O1)],comunica(K1,O,O1),S2)
asercion(F)
F=a(K1,implies(libro(K1),e(O,and(hombre(O),e(O1,and(profesor(O1),
comunica(K1,O,O1)))))))

2.3.2 Ejercicios

1. Incluir las reglas necesarias para que la gramática sea capaz de analizar las frases:

   (k) "Juan da un libro a Maria"

   (l) "El hombre consigue el ascenso en Seat"

2. Tanto los nombres comunes como una gran parte de los adjetivos se traducen como predicados unarios, así:

   (m) "El hombre bueno"

   se representaría como:

   ```
   e(X, and(hombre(X), bueno(X)))
   ```

   Añadir las reglas suficientes a la gramática para conseguir la representación de los adjetivos que complementan a los nombres comunes.

3. Los verbos copulativos (ser, estar) son un nexo de unión entre sujeto y atributo, conseguir una representación lógica para las siguientes frases:

   (n) "Juan es parecido a su hermano"

   (ñ) "Juan está enfadado con María"

2.4 Restricciones semánticas de selección de argumentos

Ya hemos visto cómo podemos declarar las restricciones de selección de tipo sintáctico, la gramática 5 introduce restricciones semánticas de selección, con el objetivo de que oraciones como:

(o) "Juan lee el hombre"

se detecten como agramaticales y no sean frases bien formadas en nuestra gramática.

Introduzcamos ahora nuevas frases con el objetivo de que la variación semántica sea más interesante.

(p) "El gato come pescado"

(q) "El perro corre por el camino"

Para poder realizar esta selección semántica de argumentos, es necesario en primer lugar establecer un repertorio de categorías semánticas que se declaren ya a nivel léxico. Por el momento estableceremos las siguientes categorías:

humano:	Juan, María, Clara, hombre, profesor
animado:	perro, gato
inanimado:	libro, pescado, bocadillo, silla
locativo:	Barcelona, camino

El primer cambio que debemos efectuar es a nivel léxico, categorizando semánticamente los nombres:

```
npr(clara, humano).
npr(maria, humano).
npr(juan, humano).
npr(barcelona,locativo).
nombre(libro, K,libro(K), inanimado).
nombre(hombre, K,hombre(K), humano).
nombre(profesor,K,profesor(K), humano).
nombre(perro, K,perro(K), animado).
nombre (gato, K,gato(K), animado).
nombre(camino, K,camino(K), locativo).
```

También hemos de explicitar a qué tipo de clase semántica deben pertenecer los argumentos verbales, es decir, que tipo de selección semántica aplica el verbo. En el caso de que la clase de un argumento verbal fuera demasiado amplia, podemos solucionarlo mediante la utilización de variables. Para guardar la información de los complementos y debido a que un verbo puede regir varios, adoptamos una representación en forma de lista del mismo modo que hemos tratado los argumentos desde el punto de vista sintáctico.

```
verbo(reir, S,[],reir(S), asem(humano, [])).
verbo(piensa,S,[arg(en,O)],pensar en(S,O),sem(humano,[X])).
verbo(habla,S,[arg(con,O),arg(de,O1)],comunica(S,O,O1), asem(humano,[humano,X])).
verbo(esta,A, [arg(en,O)], locativo(S,O), asem(X, [locativo])).
verbo (lee,S, [arg (O)], leer(S, O), asem(humano, [inanimado])).
verbo(corre,S, [arg(en,O)], correr(S,O), asem(animado, [locativo])).
verbo(come,S, [arg(en,O)], comer(S,O), asem(animado, [inanimado])).
```

Esta información debe también incluirse a nivel sintagmático para ello modificaremos las reglas relativas a verbos y nombres:

```
1 verb(S,A,F,Sem) --> [W], {verbo(W,S,A,F,Sem)}.
2 npr(W, Sem) --> [W], {npropio(W, Sem)}.
3 n(F, Sem)--> [W], {nombre(W,K,F, Sem)}.
```

Por ultimo hemos de modificar las reglas de la gramática donde participan estas categorías.

```
1 sn(K,S2,F, Sem)-> det(K,S1,S2,F),n(K,S1, Sem).
2 sn(K,F,F, Sem)-->npr(K, Sem).
3 compl([], S,S, [])--> [].
4 compl([arg(X,K) | Y],S1,S,[Sem | Sem2]) --> prep(X), sn(K,S2,S, Sem),
5  compl(Y,S1,S2, Sem2).
6
7 asercion(S)-> sn(K,S2,S, Sem1), verb(K,X,S1, asem(Sem1,Listasem)),
8  compl(X,S1,S2,[Listasem]).
```

Gramática 5

Esta es la quinta gramática completa.

```
1 analisis(F,X,Y):- asercion(F,X,Y).
2
3 asercion(F) -->
```

```
4     sn(K,S2,F,Sem1), verb(K,X,S1,asem(Sem1,Sem2)), compl(X,S1,S2,Sem2).
5
6   compl([],S,S,[])--> [].
7   compl([arg(X,K)|Y],S1,S,[Sem1|Sem2])--> prep(X), sn(K,S2,S,Sem1),
8                                                     compl(Y,S1,S2,Sem2).
9   compl([arg(nulo,K)|Y],S1,S,[Sem1|Sem2])-->
10                                  sn(K,S2,S,Sem1), compl(Y,S1,S2,Sem2).
11
12  sn(K,F,F,Sem)--> npr(K,Sem).
13  sn(K,S2,F,Sem)--> det(K,S1,S2,F),n(K,S1,Sem).
14
15  verb(S,A,F,AS)--> [W],{verbo(W,S,A,F,AS)}.
16  npr(W,S)--> [W],{npropio(W,S)}.
17  n(K,F,S)--> [W],{nombre(W,K,F,S)}.
18  det(K,S1,S2,F)--> [W],{determ(W,K,S1,S2,F)}.
19  prep(W)--> [W],{prepo(W)}.
20
21  npropio(clara,humano).
22  npropio(maria,humano).
23  npropio(juan,humano).
24  npropio(barcelona,locativo).
25  nombre(libro,K,libro(K),inanimado).
26  nombre(hombre,K,hombre(K),humano).
27  nombre(profesor,K,profesor(K),humano).
28  nombre(perro,K,profesor(K),animado).
29  nombre(gato,K,gato(K),animado).
30  nombre(camino,K,camino(K),locativo).
31  nombre(pescado,K,pescado(K),inanimado).
32  determ(el,K,S1,S2,e(K,and(S1,S2))).
33  determ(un,K,S1,S2,e(K,and(S1,S2))).
34  determ(los,K,S1,S2,a(K,implies(S1,S2))).
35  determ(todo,K,S1,S2,a(K,implies(S1,S2))).
36  prepo(en).
37  prepo(con).
38  prepo(de).
39
40  verbo(rie,S,[],reir(S),asem(humano,[])).
41  verbo(piensa,S,[arg(en,O)],pensar_en(S,O),asem(humano,[_])).
42  verbo(habla,S,[arg(de,O),arg(con,O1)],comunica(S,O,O1),asem(humano,[_,humano])).
43  verbo(habla,S,[arg(con,O),arg(de,O1)],comunica(S,O1,O),asem(humano,[humano,_])).
44  verbo(esta,S,[arg(en,O)],locativo(S,O),asem(_,[locativo])).
45  verbo(lee,S,[arg(nulo,O)],leer(S,O),asem(humano,[inanimado])).
46  verbo(corre,S,[arg(en,O)],correr(S,O),asem(animado,[locativo])).
47  verbo(come,S,[arg(en,O)],comer(S,O),asem(animado,[locativo])).
48  verbo(come,S,[arg(nulo,O)],comer(S,O),asem(animado,[inanimado])).
```

2.4.1 **Ejercicios**

1. Obtener una fórmula lógica etiquetada con papeles temáticos. Por ejemplo:

Un hombre come un bocadillo

una posible representación:

```
e(X, and(hombre(X), e(Y, and(bocadillo(Y), come(X,Y)))),
acción (comer (agente(hombre), paciente(bocadillo))))
```

2. Construir una estructura que clasifique de forma jerárquica las categorías semánticas utilizadas, de modo que podamos generalizar comportamientos verbales sin utilizar variables en la descripción léxica.