Implementación de una Gramática de Cláusulas Definidas en Prolog

Fco. Javier Bueno

1. Introducción

Una de las aplicaciones más extendidas de la Inteligencia Artificial se centra en el Procesamiento de Lenguaje Natural. Éste puede ser entendido como el uso de ordenadores para reconocer y usar información expresada en forma de lenguaje humano. De este modo, se han desarrollado algunas aplicaciones prácticas tales como bases de datos y sistemas de ayuda que aceptan preguntas en lenguaje natural, resúmenes automáticos de textos de tipo técnico-científico o sistemas guiados por voz. Todas ellas se basan en al análisis de estructuras sintácticas válidas como proceso fundamental sobre el que se desarrolla la aplicación concreta.

Dado que una lengua se puede estructurar en cinco niveles, como son, fonología (sonidos de las palabras), morfología (formación de palabras), sintaxis (formación de oraciones), semántica (significado de las oraciones) y pragmática (uso de la lengua en un contexto dado), el análisis sintáctico se sitúa en el nivel intermedio. Dicho análisis permite estudiar la relación existente entre las distintas palabras que forman una oración y ver si se ajustan a algún tipo de patrón que se repite en el lenguaje estudiado.

Chomsky introduce en 1957 el concepto de *gramática generativa* que permite describir oraciones por medio de reglas constructivas. Por ejemplo, las reglas de la Figura 1 pueden generar un conjunto de oraciones entre la que se incluye la mostrada en la Figura 2.

$O \rightarrow GN\ GV$
$O \rightarrow GV$
$GN \rightarrow Det\ N$
$GV \rightarrow V$
$GV \rightarrow V\ GN$
$GP \rightarrow Prep\ GN$
$Det \rightarrow este,ese,un,el,la$
$Prep \rightarrow en$
$N \rightarrow hombre,libro,vuelo,comida,tren$
$V \rightarrow incluye,lee$

Figura 1: Ejemplo de reglas gramaticales.

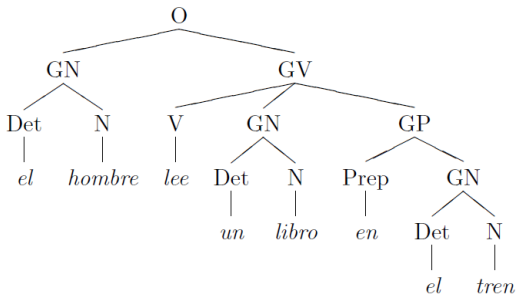

Figura 2: Ejemplo de análisis sintáctico.

1.1. Gramáticas Libres de Contexto

Uno de los modelos matemáticos más utilizados para modelar el lenguaje natural es el de las Gramáticas Libres de Contexto (*Context Free Grammars* o CFG). Una gramática libre de contexto consiste, por un lado, en un conjunto de reglas que expresan los modos en los que los símbolos lingüísticos pueden agruparse formando categorías gramaticales y, por otro, un lexicón que contiene dichos símbolos.

Las reglas (R) pueden agruparse de forma jerárquica para definir determinados tipos de categorías gramaticales y, además combinarse entre sí para formar nuevas estructuras lingüísticas constituyendo una gramática generativa o de constituyentes tal y como la definida por Chomsky.

Los símbolos usados en las gramáticas CFG pueden dividirse en dos clases: terminales y no terminales. Los símbolos terminales (T) corresponden con las palabras del lenguaje que se desea modelizar y que están recogidas en el lexicón. Los símbolos no-terminales (N) expresan agrupaciones o generalizaciones de símbolos terminales.

Para formar una gramática CFG además es necesario contar con un símbolo no-terminal inicial (S) y un conjunto de reglas de la forma $X \rightarrow \gamma$, donde X es un símbolo no-terminal y γ es una secuencia de símbolos terminales y/o no terminales o incluso puede estar vacía. De ese modo la gramática puede expresarse como $G =< T, N, S, R >$ que genera un lenguaje formal L.

En Procesamiento del Lenguaje Natural se suele distinguir un subgrupo (P) dentro de los símbolos no-terminales ($P \subset N$) denominado pre-terminales, que en la primera derivación posible dan lugar a los símbolos terminales.

Un ejemplo de gramática puede ser el mostrado en la Figura 3. Los símbolos no-terminales son en ese caso: O (oración), GN (Grupo Nominal), GV (Grupo Verbal); los pre-terminales son: Det (Determinante), Prep (Preposición), N (Nombre), V (Verbo); y los símbolos terminales son los símbolos *este, ese, un, en, el, la, hombre, libro, vuelo, comida, incluye, lee, tren* que constituyen el lexicón. El símbolo O constituye a su vez el símbolo no-terminal inicial.

El uso habitual de este tipo de gramáticas es doble: por un lado, generar nuevas oraciones pertenecientes al lenguaje L (denominadas derivaciones) y, por otro, asignar una estructura gramatical a una oración dada. Este último caso es el que nos interesa en esta práctica.

$G =< T, N, S, R >$
$T = \{este, ese, un, en, el, la, hombre, libro, vuelo, comida, incluye, lee, tren\}$
$N = \{O, GN, N, GV, Det, Prep, N, V\}$
$S = \{O\}$
$R = \{$
$O \rightarrow GN\ GV$
$O \rightarrow GV$
$GN \rightarrow Det\ N$
$GV \rightarrow V$
$GV \rightarrow V\ GN$
$GP \rightarrow Prep\ GN$
$Det \rightarrow este/ese/un/el/la$
$Prep \rightarrow en$
$N \rightarrow hombre/libro/vuelo/comida/tren$
$V \rightarrow incluye/lee$
$\}$

Figura 3: Ejemplo de Gramática Libre de Contexto (CFG).

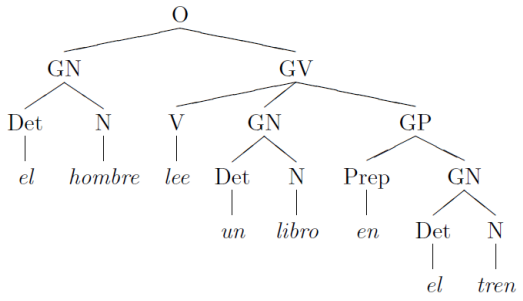

Figura 4: Ejemplo de árbol de constituyentes.

Las estructuras gramaticales se suelen representar por medio de árboles de constituyentes de modo que cada una de las subestructuras que conforma una oración quedan explicitadas de forma jerárquica. En la Figura 2 se muestra un ejemplo de árbol sintáctico de una oración generada por la gramática mostrada en la Figura 3.

Las Gramáticas Libres de Contexto se usan ampliamente en el Procesamiento de Lenguaje Natural por dos razones. La primera es que dan cuenta de organización interna de las oraciones por medio de las categorías gramaticales y la segunda es que pueden manejar estructuras recursivas. La recursividad ocurre cuando la regla que define un símbolo no-terminal incluye a dicho símbolo. Un ejemplo es el siguiente:

a) El perro persiguió al gato.

b) La niña pensó que el perro persiguió al gato.

c) El mayordomo dijo que la niña pensó que el perro persiguió al gato.

En la Figura 5 se muestra el árbol sintáctico de la última oración donde se puede apreciar que existe una estructura básica (O) que se repite de forma recursiva en la oración final.

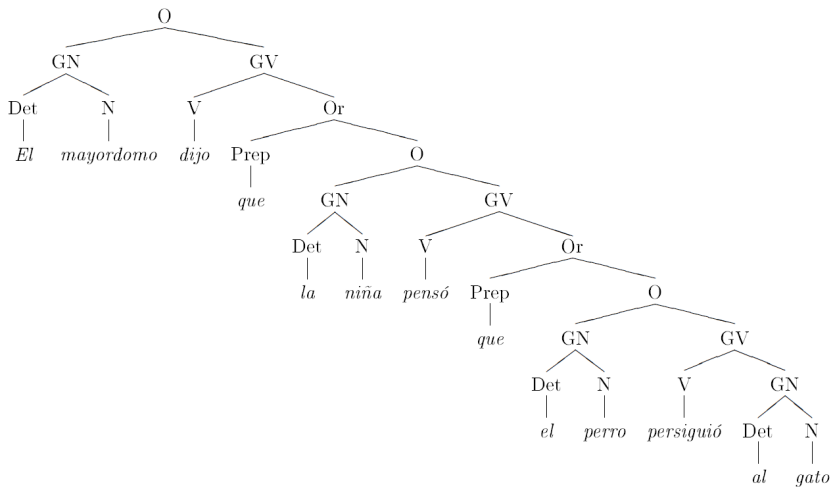

Figura 5: Ejemplo de oración en la que se repite de forma recursiva la subestructura *O*.

1.2. Gramáticas de Cláusulas Definidas

Uno de los lenguajes de programación más utilizados en Procesamiento de Lenguaje Natural es Prolog (Programación Lógica) implementado por Colmerauer a principio de los 70' con el propósito inicial de traducir entre lenguajes naturales. Su característica principal es que una vez que el programador ha definido cuál es el problema, el sistema aplica la deducción lógica para encontrar respuestas.

De ese modo, un programa en Prolog consiste en la especificación de una serie de hechos, la definición de una serie de reglas y el planteamiento de preguntas acerca de los objetos, sobre los que se han definido los hechos y las reglas, y sus relaciones.

Por tanto, es posible especificar gramáticas libres de contexto en Prolog ya que se basa en reglas y hechos acerca de objetos, es decir se pueden establecer reglas gramaticales (R), basadas en hechos (N), sobre un lexicón (T) compuesto de palabras (objetos).

La forma tradicional de especificar este tipo de gramáticas en Prolog se ha basado en las Gramáticas de Clausulas Definidas (GCD o DCG en inglés). Una cláusula definida tiene la forma

$P \Leftarrow Q_1 \& ... \& Q_n$

donde P es la cabecera y $Q_1, ..., Q_n$ forman el cuerpo de la cláusula. Aquella debe leerse como 'P es cierto si Q_1 y ... y Q_n son ciertos'.

Así, una regla libre de contexto del tipo:

$X \rightarrow \alpha_1 \ ... \ \alpha_n$

puede traducirse a la cláusula definida

$x(S_0, S_n) \Leftarrow \alpha_1(S_0, S_1) \& ... \& \alpha_1(S_{n-1}, S_n)$

donde las variables S_i representan posiciones de la cadena de texto que forma la oración. Por ejemplo, la regla libre de contexto

$O \rightarrow GN \ GV$

se puede traducir a la cláusula definida

$O(S_0, S_2) \Leftarrow GN(S_0, S_1) \& GV(S_1, S_2)$.

cuyo significado es 'Existe una oración entre S_0 y S_2 si existe un grupo nominal entre S_0 y S_1, y existe un grupo verbal entre S_1 y S_2'. Para completar la gramática habría, además, que definir qué es un *grupo_nominal* y un *grupo_verbal*, así como las reglas para formar cada uno de dichos símbolos no-terminales.

En Prolog existen dos formas de especificar las cláusulas definidas. La traducción literal de la cláusula anterior sería:

```
oracion(S0,S) :- grupo_nominal(S0,S1), grupo_verbal(S1,S).
```

pero, por fortuna, existe una notación simplificada que Prolog admite, de tal modo que la regla anterior quedaría de forma muy parecida a como se especifican las reglas CFG, esto es:

```
oracion --> grupo_nominal, grupo_verbal.
```

2. Creación de una gramática sencilla

El punto de partida para el desarrollo de la práctica consiste en la siguiente gramática:

```
% Reglas gramaticales
oracion --> g_nominal, g_verbal.

g_nominal --> nombre.
g_nominal --> determinante, nombre.
g_nominal --> nombre, adjetivo.

g_verbal --> verbo.
g_verbal --> verbo, g_nominal.
g_verbal --> verbo, adjetivo.

%Diccionario
determinante --> [el].
determinante --> [la].
determinante --> [un].
determinante --> [una].

nombre --> [hombre].
nombre --> [mujer].
nombre --> [juan].
nombre --> [maría].
nombre --> [manzana].
nombre --> [gato].
nombre --> [ratón].
nombre --> [alumno].
```

```
nombre --> [universidad].

verbo --> [ama].
verbo --> [come].
verbo --> [estudia].


adjetivo --> [roja].
adjetivo --> [negro].
adjetivo --> [grande].
adjetivo --> [gris].
adjetivo --> [pequeño].
```

que permite decidir si alguna de estas oraciones se ajustan a la gramática definida:

1. *El hombre come una manzana.*
2. *La mujer come manzanas.*
3. *María come una manzana roja.*
4. *Juan ama a María.*
5. *El gato grande come un ratón gris.*
6. *Juan estudia en la Universidad.*
7. *El alumno ama la Universidad.*
8. *El gato come ratones.*
9. *La manzana come un gato.*
10. *La Universidad es grande.*

Para comprobarlo, es necesario cargar la gramática en Swi-Prolog e introducir consultas del tipo:

```
?- oracion([el,hombre,come,una,manzana],[]).
```

donde es necesario comprobar que todas las palabras de la oración estén en minúsculas (tal y como han sido definidas en el diccionario).

Si la oración es válida según la gramática, Prolog devolverá una respuesta afirmativa. En caso contrario será necesario analizar cuál es la causa del rechazo.

Del mismo modo, se puede preguntar si un fragmento dado de una oración se admite como grupo nominal, grupo verbal o cualquiera de los constituyentes de la gramática.

```
?- g_verbal([come,una,manzana],[]).
```

2.1. Ejercicio

Comprobar si la gramática y el diccionario propuestos en este apartado son capaces de validar todas las frases anteriores (para que el intérprete Prolog admita tildes y eñes es necesario **guardar el archivo** con codificación **ANSI**).

En caso de que alguna de las frases no sea admitida, modificar la gramática y el diccionario para que la valide.

3. Estructura sintáctica

La gramática anterior solamente indica si una oración se ajusta a la gramática definida o no, pero no permite analizar el árbol de constituyentes (árbol sintáctico) de la misma. Para ello, es necesario incluir argumentos en la definición de las reglas gramaticales de modo Prolog devuelva una estructura de datos que recoja la estructura gramatical validada por la gramática.

Un ejemplo de las nuevas reglas gramaticales es el siguiente:

```
oracion(o(GN,GV)) --> g_nominal(GN), g_verbal(GV).

g_nominal(gn(N)) --> nombre(N).
g_nominal(gn(D,N)) --> determinante(D), nombre(N).
...

determinante(det(X)) --> [X],{det(X)}.
det(el).
det(la).
...

nombre(n(X)) --> [X],{n(X)}.
n(hombre).
n(mujer).
...
```

En este caso, la consulta debe realizarse del siguiente modo para que Prolog devuelva la estructura de la oración:

```
?- oracion(X,[el,hombre,come,una,manzana],[]).
```

y el resultado es:

```
o(gn(det(el),n(hombre)),gv(v(come),gn(det(una),n(manzana))))
```

Por último, para visualizar la estructura sintáctica en forma de árbol es necesario cargar la aplicación **draw.pl** desde el archivo de la gramática. Ello se logra mediante el uso del predicado

```
:- consult(draw).
```

En la línea de comandos de Prolog es necesario indicar que deseamos representar la estructura devuelta:

```
?- oracion(X,[el,hombre,come,la,manzana],[]), draw(X).
```

y el resultado queda de este modo:

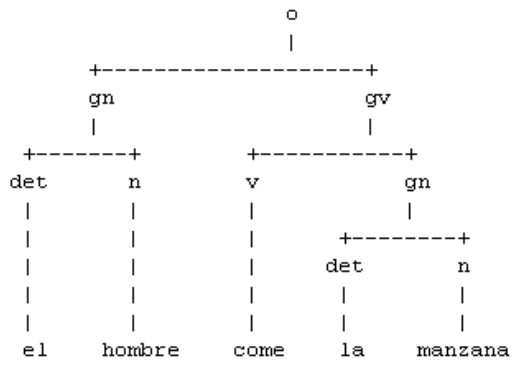

Figura 6: Árbol de constituyentes.

(El programa **draw.pl** se encuentra en la plataforma de la asignatura).

3.1. Ejercicio

Modificar la gramática y el diccionario propuestos en el apartado anterior para que representen gráficamente la estructura gramatical de todas las frases de la página 6, además de las siguientes:

1. *Juan y María comen una manzana con un tenedor y un cuchillo.*
2. *María hace la práctica de Juan.*
3. *Mi casa está muy lejos de Madrid.*
4. *Ana fue ayudada por un enfermero.*
5. *Nunca llueve en el desierto.*
6. *Todos se comportaron bien en clase.*
7. *Esta noche saldré por Alcalá.*
8. *La esperanza de vida de un niño depende de su lugar de nacimiento.*