Capítulo 3

El *Chart Parser*

3.1. Introducción

El analizador sintáctico que se usa en esta tesis es el así llamado *Chart Parser* sugerido por [Dá94] como el más apropiado para ser implantado en la Teoría de Tipos de Martin-Löf por no requerir búsqueda en reversa o *backtracking* y por ser una técnica conocida. A continuación se da una descripción muy general del *chart parser* y su funcionamiento.

3.2. La Gramática

Para examinar cómo puede ser procesada la estructura sintáctica de una oración debemos considerar dos cosas: la **gramática**, que es una especificación formal de las estructuras permisibles en el lenguaje y la **técnica del análisis sintáctico**, que, como su nombre lo dice, es el método de analizar una oración para determinar su estructura de acuerdo a la gramática [All95].

El camino más común de representar cómo una oración se descompone en sus "subpartes", y cómo esas "subpartes" se descomponen a su vez, es por medio de un **árbol**. En general, un árbol es considerado como una forma especial de grafo, consistiendo así de nodos etiquetados conectados por ligas. El nodo en la cima se le llama **raíz** del árbol, mientras que a los nodos que le siguen se les llama **hojas**. Y para terminar, un **ancestral** de un nodo N se define como el padre de N, o el padre del padre de N, y así sucesivamente. Un nodo es **dominado** por sus nodos ancestrales. Entonces el nodo raíz domina a todos los otros nodos del árbol. [All95]

1.	$S \rightarrow NP\ VP$	*12.*	$IV \rightarrow ate$
2.	$NP \rightarrow DET\ NB$	*13.*	$NOUN \rightarrow bread$
3.	$NB \rightarrow NOUN$	*14.*	$DET \rightarrow a$
4.	$NB \rightarrow NOUN\ REL$	*15.*	$DET \rightarrow the$
5.	$REL \rightarrow WH\ VP$	*16.*	$NOUN \rightarrow man$
6.	$VP \rightarrow IV$	*17.*	$WH \rightarrow who$
7.	$VP \rightarrow TV\ NP$	*18.*	$WH \rightarrow that$
8.	$VP \rightarrow DV\ NP\ PP$	*19.*	$DET \rightarrow he$
9.	$VP \rightarrow SV\ S$	*20.*	$NP \rightarrow John$
10.	$PP \rightarrow P\ NP$	*21.*	$P \rightarrow to$
11.	$TV \rightarrow ate$		

Cuadro 3.1: La gramática.

Para construir una estructura de una oración, y siguiendo la explicación de [All95], se debe saber cuáles estructuras son válidas en inglés. Así, un conjunto de **reglas de producción** describe las estructuras de árbol que son permitidas. Estas reglas nos dicen que un cierto símbolo puede ser expandido en el árbol por una secuencia de otros símbolos. Por otro lado, la gramática que usamos en este proyecto fue extraída en su mayoría del código original de [GM89b] en *Prolog* y la presentamos en la tabla 3.2. En esta gramática, la regla 1 dice que una oración *S* puede consistir de una **frase nominal** *NP* seguida por una **frase verbal** *VP*. De igual forma, la regla 2 dice que una frase nominal *NP* puede consistir de un **determinante** *DET* seguido de una *NB*. Las reglas 3 y 4 dicen que una *NB* puede consistir de un **nombre** *NOUN* o de un nombre *NOUN* seguido de un *REL*. Las reglas 11–21 definen las categorías de las palabras. Por otro lado, las gramáticas que consisten en su totalidad de reglas con un símbolo simple del lado izquierdo se les denomina **gramáticas libres del contexto** (CFGs de sus siglas en inglés) [All95]. Los símbolos de la gramática que ya no pueden ser descompuestos, tales como los de las reglas 11–21, se les llama **símbolos terminales**; y por el contrario, a los otros símbolos se les llama **símbolos no terminales**. Finalmente, los símbolos gramaticales que describen categorías de palabras, (por ejemplo, *IV* o *NOUN*) reciben el nombre de **símbolos léxicos**.

Las gramáticas tienen un símbolo especial llamado el símbolo inicial

[All95]. En esta tesis ese símbolo inicial será siempre S. En [All95] se dice que una gramática **deriva** una oración si hay una secuencia de reglas que permitan producir el símbolo inicial en la oración. Por ejemplo, la gramática de la tabla 3.2 deriva la oración *"John ate a bread"* y se puede comprobar con la siguiente secuencia:

S	
$\Rightarrow$ *NP VP*	(produciendo *S*)
$\Rightarrow$ *John VP*	(produciendo *NP*)
$\Rightarrow$ *John TV NP*	(produciendo *VP*)
$\Rightarrow$ *John ate NP*	(produciendo *TV*)
$\Rightarrow$ *John ate DET NB*	(produciendo *NP*)
$\Rightarrow$ *John ate a NB*	(produciendo *DET*)
$\Rightarrow$ *John ate a NOUN*	(produciendo *NB*)
$\Rightarrow$ *John ate a bread*	(produciendo *NOUN*)

Una gramática *generativa* se define en [Dá94] formalmente como:

$$G = \{T, NT, S, P\}$$

donde

T	=	Conjunto de símbolos terminales.
NT	=	Conjunto de símbolos no terminales.
S	=	Símbolo inicial que pertenece a NT.
P	=	Conjunto de reglas de producción.

Una regla de producción tiene la forma

$$\alpha \rightarrow \beta$$

donde α y β son cadenas de símbolos que pertenecen a T o a NT.

3.3. El Análisis Sintáctico

En [All95] se establece que hay dos procesos importantes basados en derivaciones, la generación de oraciones y el análisis sintáctico. El primero, la **generación de oraciones**, usa derivaciones para construir oraciones legales. Un generador simple podría ser implementado escogiendo aleatoriamente reglas de producción [All95], comenzando a partir del símbolo inicial S, hasta

que se tenga una secuencia de palabras. El ejemplo de la sección anterior muestra que la oración *"John ate a bread"* puede ser generada a partir de la gramática. El segundo proceso basado en derivaciones es el **análisis o reconocimiento sintáctico** conocido en inglés como ***parsing***, el cual, dada una gramática, identifica la estructura de oraciones. Aquí tenemos dos métodos básicos de búsqueda [All95]. La **estrategia *top-down*** (o de lo general a lo particular) que empieza con el símbolo inicial S y busca caminos diferentes para producir los símbolos hasta que la secuencia de entrada es generada, o hasta que todas las posibilidades hayan sido exploradas. El ejemplo de la sección anterior demuestra que *"John ate a bread"* es una oración válida mostrando la derivación que pudo ser encontrada por este proceso.

En una **estrategia *bottom-up*** uno puede iniciar con las palabras en la oración y usar las reglas de producción en reversa para reducir la secuencia de símbolos hasta que consista solamente de S [All95]. La parte izquierda de cada regla es usada para producir el símbolo en la parte del lado derecho. Un análisis *bottom-up* posible de la oración *"John ate a bread"* es

$\Rightarrow$ *NP ate a bread*	(produciendo *John*)
$\Rightarrow$ *NP TV a bread*	(produciendo *ate*)
$\Rightarrow$ *NP TV DET bread*	(produciendo *a*)
$\Rightarrow$ *NP TV DET NOUN*	(produciendo *bread*)
$\Rightarrow$ *NP TV DET NB*	(produciendo *NOUN*)
$\Rightarrow$ *NP TV NP*	(produciendo *DET NB*)
$\Rightarrow$ *NP VP*	(produciendo *TV NP*)
$\Rightarrow$ *S*	(produciendo *NP VP*)

Y si se mantiene un registro del proceso de reconocimiento, ya sea en *bottom-up* o *top-down*, se obtendrá un árbol.

Es así que, dada una gramática y una secuencia de palabras, la actividad en el análisis es el proceso de verificar cuándo una secuencia de palabras es una expresión bien formada generada por la gramática y cuándo no [Dá94]. Esperamos también, como característica adicional del proceso, la estructura o estructuras de la oración. En [Dá94] se establece que para una gramática libre del contexto, esta información estructural generalmente se provee en la forma de un árbol. Dicho árbol contendrá toda la información concerniente a la composición de la oración, de acuerdo con las reglas de la gramática que la genera [Dá94].

3.4. El *Bottom-Up Chart Parser*

La diferencia principal entre reconocimiento *bottom-up* y *top-down* es la forma como se usan las reglas gramaticales [All95]. Por ejemplo, considerando la regla de producción

$$NP \rightarrow DET\ NB$$

en un sistema top-down se usa la regla para encontrar una *NP* buscando la secuencia *DET NB*. En contraste, en un reconocimiento *bottom-up* se usa la regla para tomar una secuencia *DET NB* que se ha encontrado e identificado como una *NP*. Por lo tanto, la operación básica en un reconocimiento *bottom-up* es tomar una secuencia de símbolos y acoplarlos con la parte derecha de las reglas [All95]. Según [All95], se puede construir un analizador sintáctico *bottom-up* simplemente formulando este proceso de acoplamiento como un proceso de búsqueda. La condición consistiría de una lista de símbolos, empezando con las palabras en la oración. Las otras condiciones podrían ser generadas explorando todas las formas posibles para

- producir una palabra por sus categorías léxicas posibles
- reemplazar una secuencia de símbolos que se acople a la parte derecha de una regla gramatical por su símbolo en el lado izquierdo.

En [Dá94] un ***chart parser*** se define como un sistema que contiene una estructura llamada ***chart*** en la cual se almacenan los registros de los constituyentes y subconstituyentes que se van encontrando en el proceso de análisis sintáctico. El sistema lee otra estructura que contiene las reglas gramaticales. Esto da como ventaja que se pueda modificar la gramática o incluso cambiarla por una nueva sin alterar el resto del sistema. Representado gráficamente, el analizador sintáctico se vería como en la figura 3.1.

Y por último, en un *chart parser* se distinguen tres funciones principales y una condición de terminación [Dá94]:

- Inicialización del *chart*.
- Regla *Bottom-Up*.
- Regla Fundamental.

La condición de terminación del proceso de un *chart parser* consiste en mantenerlo trabajando sobre la expresión hasta que la aplicación de la regla fundamental ya no añada más arcos al *chart* (siempre y cuando la gramática no contenga reglas de la forma $S \rightarrow S$, que evitarían la terminación) [Dá94].

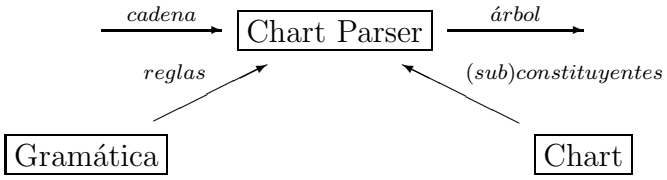

Figura 3.1: Representación esquemática de un *chart parser*.

3.5. El *Chart*

Como ya hemos establecido en líneas anteriores, el *chart* es una estructura en la cual se almacenan los registros de los constituyentes y subconstituyentes que se van encontrando en el proceso de análisis sintáctico. Es un grafo acíclico dirigido[1] donde cada arco significa un constituyente de la oración [Dá94] y cada nodo es el lugar de dicho constituyente. Para representar el *chart*, Gazdar y Mellish [GM89b] usan dos formas de registros —una de la forma $\alpha \rightarrow \beta$ y la otra codificada. No obstante hemos cambiado un poco su sintaxis[2] con propósito de consistencia. Así, cada arco es representado con una de las tres clases que se enuncian a continuación:

$$\begin{aligned} A \rightarrow &\quad BC \bullet \\ A \rightarrow &\quad B \bullet C \\ A \rightarrow &\quad \bullet BC \end{aligned}$$

donde el "punto" divide la parte derecha de las reglas de producción en hipótesis a probar e hipótesis probadas. Por ejemplo, la primera forma del registro ocurre solamente en arcos que regresan al mismo nodo inicial, y denota la hipótesis de que una A puede ser encontrada cubriendo una subcadena que empieza en el nodo en cuestión y que es una secuencia de B y C. Un registro de esta forma indica que tal hipótesis ha sido hecha pero que no necesariamente ha sido verificada parcialmente, todavía. La segunda y tercera

[1] Hablando en sentido estricto, este grafo **no es acíclico** porque permite arcos vacíos que regresan al mismo nodo donde se empezó. Sin embargo, queda prohibido hacer otra clase de ciclos que no sean al mismo nodo [GM89b].

[2] En esta versión de prolog, las etiquetas de punto tienen una lectura invertida a su implementación sin razón aparente, lo cual puede causar confusión.

formas denotan la misma hipótesis, pero indican que dicha hipótesis ha sido parcial o totalmente confirmada, respectivamente.

En general, la manera de interpretar el lado derecho estas reglas de producción se ilustra en el esquema siguiente:

$$\text{hipótesis a probar} \hookleftarrow \bullet \hookrightarrow \text{hipótesis probada}$$

Al conjunto de registros de este tipo se le llama en [GM89b] ***chart* activo** o simplemente *chart*, y a la familia de analizadores sintácticos que hacen uso de ellos se les denomina **analizadores *chart***, o universalmente, ***chart parsers***. De la misma forma, las etiquetas que representan hipótesis no conformadas son conocidas como **arcos activos** y aquellas que representan hipótesis confirmadas (con el punto al inicio) se les llama **arcos inactivos**. Finalmente, cuando usamos el "punto" en las etiquetas se les denomina **reglas punto** o *dotted rules*, [GM89b] en inglés.

Para propósitos de implantación y adaptando la sintaxis usada en [Dá94] para etiquetar los arcos, describiremos a continuación la codificación de las reglas punto en nuestro proyecto.

$$< start, finish, label, toFind, found >$$

donde

start	es el nodo inicial
finish	es el nodo terminal
label	es la categoría sintáctica construida o en proceso de construcción —parte izquierda de las reglas punto $\alpha \rightarrow \beta$
toFind	son listas de constituyentes sintácticos por probar —lado izquierdo del punto
found	son listas de constituyentes sintácticos probados —lado derecho del punto

Por otro lado, un arco puede ser **activo**, si contiene una recuperación parcial de la categoría sintáctica *label* (*toFind* no es vacío) o **inactivo**, si no hay más componentes que buscar (*toFind* es vacío).

Por ejemplo, los registros de arcos,

$$< a, b, A \rightarrow B \bullet C >$$

y

$$< i, j, O \rightarrow \bullet PQ >$$

serían representados respectivamente como

$$\begin{aligned} < start >= & \quad a \\ < finish >= & \quad b \\ < label >= & \quad A \\ < toFind >= & \quad < B > \\ < found >= & \quad < C > \end{aligned}$$

y

$$\begin{aligned} < start >= & \quad i \\ < finish >= & \quad j \\ < label >= & \quad O \\ < toFind >= & \quad <> \\ < found >= & \quad < P, Q > \end{aligned}$$

3.6. La Regla Fundamental

La esencia de un *Chart Parser* es la aplicación repetida de una regla conocida como la **regla fundamental** del reconocimiento *chart* [Dá94], y consiste en lo siguiente: si un arco activo encuentra un arco inactivo de la categoría deseada, entonces meter un nuevo arco al chart que extienda ambos arcos activo e inactivo [GM89b]. En [GM89b] se expresan formalmente como sigue:

> **Regla Fundamental**
> Si el *chart* contiene arcos $< i, j, A \rightarrow BW_1 \bullet W_2 >$ y $< j, k, B \rightarrow \bullet W_3 >$, donde A y B son categorías y W_1, W_2 y W_3 son secuencias (posiblemente vacías) de categorías o palabras, entonces añadir el arco $< i, k, A \rightarrow W_1 \bullet W_2 B >$ al *chart*.

En general, la aplicación de la regla fundamental se vería gráficamente como en la figura 3.2. La flecha punteada representa el nuevo arco que se añade bajo esas condiciones.

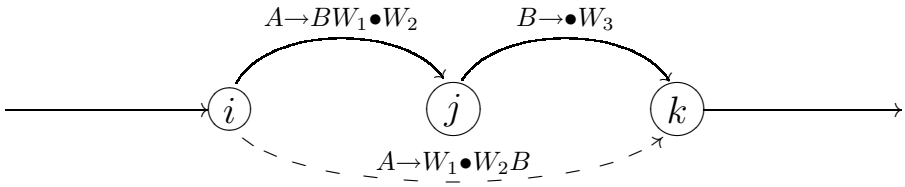

Figura 3.2: Representación esquemática de la regla fundamental.

3.7. Inicialización del *Chart*

Para la aplicación de la regla fundamental del *chart parser* es necesario que existan al menos un arco activo y otro inactivo. En este proceso se construyen los nodos y se asegura de que el *chart* inicial contenga los primeros arcos inactivos a ser encontrados.

Básicamente consiste en buscar palabras en el léxico y determinar la categoría a la que pertenecen. El resultado de esta búsqueda se sitúa directamente en el *chart* como arcos inactivos [GM89b]. Sin embargo, esto no es suficiente para aplicar la regla fundamental. Se requiere que haya también arcos activos que encuentren los inactivos. Esto se logra aplicando la regla *bottom-up* cada vez que se introduce un arco inactivo al *chart*.

3.8. La Regla *Bottom-Up*

Un principio simple enunciado por Gazdar y Mellish [GM89b] para asegurar que nuevos arcos activos sean añadidos, de tal forma que pueda resultar algo de la aplicación de la regla fundamental, es el siguiente: "cada vez que se añada un nodo inactivo de categoría C al *chart*, añadir un nuevo arco activo vacío, empezando desde el mismo nodo, para cada regla en la gramática que requiera un constituyente de categoría C como su hija por la izquierda". Un *arco activo vacío* se define [GM89b] como un arco activo que todavía tiene que encontrar alguno de sus componentes y por lo tanto empieza y termina en el mismo nodo. La regla *bottom-up* se encuentra formalmente expresada en [GM89b] como sigue:

> **Regla *Bottom-up***
> Si se añade el arco $< i, j, C \rightarrow \bullet W_1 >$ al *chart*, entonces para cada regla en la gramática de la forma $B \rightarrow CW_2$, añadir el arco $< i, i, B \rightarrow CW_2\bullet >$ al *chart*.

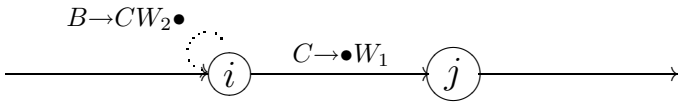

Figura 3.3: Representación esquemática de la regla *bottom-up*.

La regla *bottom-up* y la aplicación de la regla fundamental es todo lo que se necesita para asegurar que todos los análisis posibles sean encontrados [GM89b]. De forma general podemos representar la aplicación de la regla *bottom-up* como se muestra en la figura 3.3. La flecha punteada representa el nuevo arco que se añade bajo esa condición.

El uso de arcos inactivos significa que el *parser* está manteniendo hipótesis a cerca de otras frases que pudieran estar presentes. Sin embargo, nada pasará con los arcos activos hasta que otras frases requeridas hayan sido encontradas (*bottom-up*) [GM89b].

3.9. Condición de Salida y Otros Cuidados

Entendido ya el proceso de reconocimiento del *chart parser*, su terminación se lograría una vez que el arco inactivo del nodo inicial al nodo final ha sido trazado. Sin embargo, como se requiere encontrar todas las definiciones estructurales posibles de la oración, se debe mantener la búsqueda hasta que la aplicación de la regla fundamental deje de producir arcos en el *chart* (siempre y cuando no haya reglas de la forma $S \rightarrow S$ directa o indirectamente, que evitarían la terminación) [Dá94].

Al hacer uso de un reconocimiento *chart parser* se deben mantener dos principios [Dá94]:

- No remover elementos del chart.
- Siempre que se vaya a añadir un nuevo arco al *chart* checar que éste no exista ya.

3.10. Consideraciones de Eficiencia

Según [All95], "los analizadores sintácticos basados en *chart* pueden ser considerablemente más eficientes que los que dependen únicamente de una búsqueda porque el mismo constituyente nunca es construido más de una vez.

Por ejemplo, una estrategia de búsqueda puramente *top-down* o *bottom-up* podría requerir hasta C^n operaciones para reconocer una oración de tamaño n, donde C es una constante que depende del algoritmo específico que se use. Incluso si C es muy pequeña, esta complejidad exponencial rápidamente hace al algoritmo obsoleto. Por otro lado, un analizador sintáctico basado en el *chart*, en el peor de los casos construiría cada constituyente posible entre cada par de posiciones posible. esto nos permite mostrar que tiene una complejidad en el peor de los casos de $K * n^3$, donde n es la longitud de la oración y K es una constante dependiendo del algoritmo. Claro que un *chart parser* involucra más trabajo en cada paso, así que K será más grande que C. Para contrastar ambos enfoques, podemos asumir que C es 10 y que K es cien veces peor, 100. Dada una oración de 12 palabras, la fuerza bruta de búsqueda podría tomar 10^{12} operaciones (que es igual a $1,000,000,000,000$), mientras que el *chart parser* tomaría $1000 * 12^3$ (que es igual a $1,728,000$ operaciones). Bajo estas suposiciones, el *chart parser* sería hasta $500,000$ veces más rápido que la búsqueda en fuerza bruta en algunos ejemplos."