Lenguajes y Autómatas

Modelos de computación

Dr. Said Polanco Martagón[1]

Universidad Politécnica de Victoria

August 5, 2021

[1]E-mail address: spolancom@upv.edu.mx

Outline

Introducción

Son muchas las tareas que puede realizar un ordenador. Dada una tarea. se pueden plantear dos preguntas. La primera es: ¿puede realizarse con un ordenador? Una vez que sabemos que la respuesta a esa pregunta anterior es afirmativa, podemos planteamos la segunda pregunta. ¿Cómo puede realizarse esa tarea? Los modelos de computación se utilizan para responder a ambas cuestiones.

En esta unidad estudiaremos tres tipos de estructuras usadas en modelos de computación, a saber, gramáticas, máquinas de estado finito y máquinas de Turing.

Las *gramáticas* se utilizan para generar las palabras de un lenguaje y decidir si una palabra cualquiera pertenece o no al lenguaje definido. Los *lenguajes formales* definidos por gramáticas proporcionan modelos para lenguajes naturales, como el español y para lenguajes de programación, como Pascal. Fortran. Prolog, C o Java. En particular, las gramáticas son de gran importancia en la teoría y construcción de compiladores.

En los procesos de modelado se utilizan varias máquinas de estado finito. Todas las máquinas de estado finito tienen un conjunto de estados, incluido el estado inicial, un alfabeto fuente, y una función de transición que a cada pareja de estado y dato de entrada le asigna el estado siguiente.

Algunas máquinas de estado finito producen un símbolo como dato de salida para cada transición; estas máquinas pueden utilizarse para modelar una gran variedad de máquinas. entre las que se incluyen las máquinas expendedoras, los semáforos, lo sumadores binarios y los reconocedores de lenguajes. También estudiaremos máquinas de estado finito que no generan datos de salida, pero tienen estados finales. Estas máquinas se utilizan con gran frecuencia en el reconocimiento de lenguajes.

Finalmente, estudiaremos el concepto de máquina de Turing. Probaremos cómo se pueden usar las máquinas de Turing para reconocer conjuntos. También demostraremos cómo se pueden utilizar las máquinas de Turing para construir funciones aritméticas.

Outline

Introducción a las gramáticas

La gramática española nos dice cuando cierta combinación de palabras es una frase válida.

Ejemplo de frase válida

El sapo escribe claramente

Ejemplo de frase no válida

Nada rapidamente matemáticas

Nota

No nos preocupa que la frase tenga sentido, porque lo que nos ocupa es solamente la **sintaxis**, o forma, en lugar de la **semántica**, o el significado.

Lenguaje natural y lenguaje formal

La sintaxis de un **lenguaje natural**, esto es, la de los lenguajes hablados es extremadamente complicada. De hecho, parece imposible especificar todas la reglas de la sintaxis de un lenguaje natural.

Las investigaciones en el área de la traducción automática de un lenguaje a otro ha dado lugar al concepto de **lenguaje formal**, que está *especificado por un conjunto de reglas finitas bien definidas.*

Se describen las frases de un lenguaje formal utilizando una gramática. El uso de la gramática es de gran ayuda cuando se trata de resolver las dos clases de problemas que aparecen con mucha frecuencia en las aplicaciones a los lenguajes de programación:

- ¿Cómo se puede determinar cuándo una combinación de palabras es una frase válida en un lenguaje formal?
- ¿Cómo se pueden generar las frases válidas de un lenguaje formal?

Ejemplo de gramática

Se describe un ejemplo de gramática que genera un subconjunto del español. Definimos este subconjunto usando una lista de reglas que describe cómo construir una frase válida. Concretamente:

1. una **frase** se compone de un **sujeto** seguido de un **predicado**;
2. un **sujeto** se compone de un **artículo** seguido de un **nombre** seguido de un **adjetivo**, o
3. un **sujeto** se compone de un **artículo** seguido de un **nombre**;
4. un **predicado** se compone de un **verbo** seguido de un **adverbio**, o
5. un **predicado** se compone de un **verbo**;
6. un artículo es *un*. o
7. un artículo es *el*;
8. un adjetivo es *grande*, o
9. un adjetivo es *hambriento*,
10. un nombre es *conejo*, o
11. un nombre es *matemático*;
12. un verbo es *come*, o
13. un verbo es *salta*;
14. un adverbio es *rápidamente*, o
15. un adverbio es *salvajemente*.

Cont. Ejemplo de gramática

A partir de las reglas anteriores, podemos formar frases realizando una serie de reemplazamientos hasta que no podamos aplicar ninguna regla más. Por ejemplo, para obtener una frase válida podemos realizar la siguiente secuencia de sustituciones:

- **frase**
- **sujeto predicado**
- **artículo nombre adjetivo predicado**
- **artículo nombre adjetivo verbo adverbio**
- el **nombre adjetivo verbo adverbio**
- el conejo **adjetivo verbo adverbio**
- el conejo grande **verbo adverbio**
- el conejo grande salta **adverbio**
- el conejo grande salta rápidamente

Outline

Definiciones

Definición 1

Un *alfabeto* (o *vocabulario*) V es um conjunto finito y no vacío, cuyos elementos se llaman *símbolos*. Una *palabra* sobre V es una cadena finita de elementos de V. La *palabra vacía* o *cadena vacía*, denotada por λ, es la cadena sin símbolos. El conjunto de todas las palabras sobre V se denota por $V*$. Un *lenguaje sobre* V es un subconjunto de $V*$.

Nótese que λ, la palabra vacía, es la cadena que no contiene símbolos, y es diferente del conjunto vacío, $\emptyset$. Por tanto, $\{\lambda\}$ es el conjunto que contiene exactamente una palabra. la palabra vacía.

Los lenguajes se pueden especificar de varias formas:

- Enumerar todas las palabras que conforman el lenguaje.
- Establecer criterios que una palabra debe satisfacer para pertenecer al lenguaje.

En esta sección se describe otro modo importante de especificar un lenguaje, mediante una gramática, como, por ejemplo, el conjunto de reglas que e dio al inicio de la sección.

Gramática

Una *gramática* proporciona un conjunto de símbolos de varios tipos y un conjunto de reglas para construir palabras.

Concretamente, una gramática consta de un **vocabulario** o **alfabeto** V, que es el conjunto de símbolos usados para obtener los elementos de un lenguaje. Algunos de los elementos del vocabulario no se pueden reemplazar por otros símbolos. Estos elementos se llaman **terminales**, y los restantes elementos del vocabulario. aquellos que pueden sustituirse por otros símbolos, se llaman **no terminales**. El conjunto de símbolos terminales y no terminales se denotan, usualmente, por T y N, respectivamente. En el alfabeto hay un elemento especial llamado **símbolo inicial**, denotado por S, que es un elemento del vocabulario por el que siempre comenzamos. Se llama **producción** de la gramática a toda regla que especifica cuándo se puede reemplazar una cadena de $v*$, el conjunto de todas las cadenas finitas de elementos del vocabulario, por otra cadena. Se denota por $z_0 \rightarrow z_1$ la producción que establece que z_0 puede reemplazarse por z_1 en una cadena.

En ejemplo de la introducción, es $T = \{un, el, conejo, matematico, salta, come, rapidamente, salvajemente\}$, y el conjunto ele no terminales es $N = \{frase, sujeto, predicado, adjetivo, articulo, nombre, verbo, adverbio\}$. El símbolo inicial es **frase**. Y se listaron las producciones de la gramática. La primera producción, escrita utilizando esta notación, es $frase \rightarrow sujeto\ predicado$.

Definición 2

Una *gramática con estructura de frases* $G = (V, T, S, P)$ consiste en un vocabulario V, un subconjunto T de V formado por los elementos terminales, un símbolo inicial S de $V - T$ y un conjunto P de producciones. El conjunto $V - T$ se denota por N. Los elementos de N se llaman elementos *no terminales*. Toda producción de P debe contener al menos un elemento no terminal en su lado izquierdo.

Ejemplo 1

Sea $G = (V, T, S, P)$, donde $V = \{a, b, A, B, 1\}$, $T = \{a, b\}$. S es el símbolo inicial y $P = \{S \to ABa,\ A \to BB,\ B \to ab,\ AB \to b\}$. G es un ejemplo de gramática con estructura de frases.

Estamos interesados en las palabras que pueden generarse mediante las producciones de una gramática con estructura de frases.

Definición 3

Sea $G = (V, T, S, P)$ una gramática con estructura de frases. Sean $w_0 = lz_0r$ (esto es, la concatenación de l, z_0 y r) y $w_1 = lz_1r$ cadenas sobre V. Sí $z0 \rightarrow z_1$, es una producción de G, decimos que w_1 *se deriva directamente* de w_0 (o que es *directamente derivable*), y escribimos $w_0 \Rightarrow w_1$. Si $w_0, w_1, \ldots, w_n$ son cadenas sobre V tales que $w_0 \Rightarrow w_1$, $w_1 \Rightarrow w_2$, $\ldots$, $w_{n-1} \Rightarrow w_n$, decimos que w_n es *derivable*, o se *deriva*, de w_0, y se denotará $w_0 \overset{*}{\Rightarrow} w_n$. La secuencia de pasos utilizada para obtener w_n a partir de w_0 se llama derivación.

Ejemplo 2

La cadena $Aaba$ se deriva directamente de ABa en la gramática del Ejemplo I. puesto que $B \rightarrow ab$ es una producción de dicha gramática. La cadena $abababa$ se deriva de ABa, puesto que
$ABa \Rightarrow Aaba \Rightarrow BBaba \Rightarrow Bababa \Rightarrow abababa$, donde se han utilizado las producciones $B \rightarrow ab$, $A \rightarrow BB$, $B \rightarrow ab$ y $B \rightarrow ab$ sucesivamente.

Definición 4

Sea $G = (V, T, S, P)$ una gramática con estructura de frases. El *lenguaje generado* por G (o el *lenguaje de* G), denotado por $L(G)$, es el conjunto de todas las cadenas de terminales que se derivan del estado inicial S. En otras palabras,

$$L(G) = \{w \in T^* | s \overset{*}{\Rightarrow} w\}$$

Ejemplo 3

Sea G la gramática con vocabulario $V = \{S, A, a, b\}$, conjunto de terminales $T = \{a, b\}$, símbolo inicial S y producciones $P = \{S \to aA, S \to b, A \to aa\}$. ¿Cuál es $L(G)$, el lenguaje generado por esta gramática?

Solución

A partir del estado injcial S, se puede derivar aA utilizando la producción $S \to aA$. También se puede usar la producción $S \to b$ para derivar b. De aA medianre la producción $A \to aa$ se deriva aaa. Puesto que no puede derivarse ninguna otra palabra utilizando las producciones, se tiene que $L(G) = \{b, aaa\}$.

Ejemplo 4

Sea G la gramática con vocabulario $V = \{S, 0, 1\}$, conjunto de te1minales $T = \{0, 1\}$, símbolo inicial S y producciones $P = \{S \to 11S,\ S \to 0\}$. ¿Cuál es $L(C)$, el lenguaje generado por esta gramática?

Solución

A partir de S, se puede derivar 0 utilizando $S \to 0$ o bien $1S$ si empleamos $S \to 11S$. A partir de $11S$, se puede derivar bien 110 o bien $1111S$. De $1111S$ se puede derivar 11110 y $111111S$. En cualquier paso de una derivación, se pueden añadir dos unos al final de la cadena o terminar la derivación añadiendo un 0 al final de la cadena. Conjeturamos que $L(G) = \{0, 110, 11110, 1111110, \dots\}$, el conjunto de todas las cadenas que comienzan con un número par de unos y terminan con un 0.

Ejercicio

Demostrar por inducción que, después n producciones las únicas cadenas de terminales generadas son aquellas que contienen $n - 1$ o menos concatenaciones de 11 seguidas por un 0.

Con frecuencia se plantea el problema de construir una gramática que genere un lenguaje dado. Los ejemplos 5, 6 y 7 describen problemas de este tipo.

Ejemplo 5

Construye una gramática con estructura de frases que genere el conjunto $\{0^n 1^n | n = 0, 1, 2, \dots\}$

Solución

Se pueden utilizar dos producciones para generar todas las cadenas que consisten en una cadena de ceros seguida de una cadena con el mismo número de unos, incluyendo la palabra vacía. La primera palabra crece formando cadenas cada vez más largas del lenguaje mediante la concatenación de un 0 al principio de la cadena y un 1 al final. La segunda producción reemplaza S por la cadena vacía. La solución es la gramática $G = (V, T, S, P)$, donde $V = \{0, 1, S\}$, $T = \{0, 1\}$, S es el símbolo inicial y las producciones son

Ejercicio

Verificar que esta gramática genera el conjunto correcto.

Ejemplo 6

Obtén una gramática con estructura de frases que genere el conjunto $0^m 1^n | m$ y m son enteros no negativos

Daremos dos gramáticas G_1 y G_2 que generan este conjunto. Este ejemplo ilustra cómo dos gramáticas distintas pueden generar el mismo lenguaje.

Solución mediante la gramática G_1

La gramática G_1 consta del alfabeto $V = \{S, 0, 1\}$, conjunto de terminales $T = \{0, 1\}$ y las producciones $S \to 0S$, $S \to S1$ y $S \to \lambda$. G_1 genera el conjunto correcto, puesto que utilizando la primera producción m veces, se colocan m ceros al inicio de la cadena, y usando la segunda producción n veces, tenemos n unos al final de la cadena.

Ejercicio

Verificar que la gramática G_1 genera el conjunto correcto

Solución mediante la gramática G_2

La gramática G_2 consta del alfabeto $V = \{S, A, 0, 1\}$, conjunto de terminales $T = \{0, 1\}$ y las producciones $S \to 0S$, $S \to 1A$, $S \to 1$, $A \to 1A$, $A \to 1$ y $S \to \lambda$.

Ejercicio

Verificar que la gramática G_2 genera el conjunto correcto

En ocasiones, un conjunto de descripción sencilla sólo se puede generar mediante una gramática complicada.

Ejemplo 7

Una gramática que genera el conjunto $\{0^n1^n2^n | n = O.1, 2, \dots\}$ es $G = (V, T, S, P)$ con $V = \{0, 1, 2, S, A, B\}$, $T = \{0, 1, 2\}$, símbolo inicial S y las producciones $S \to 0SAB$, $S \to \lambda$, $BA \to AB$, $0A \to 01$, $1A \to 11$, $1B \to 12$, $2B \to 22$.

Ejercicio

Verificar que la gramática G genera el conjunto correcto

Outline

Introducción

Las gramáticas con estructura de frase se pueden clasificar de acuerdo con el tipo de producciones que utilicen. A continuación describiremos el esquema de clasificación introducido por Noam Chomsky. En una sección posterior se analizará que los diferentes tipos de lenguajes definidos con este esquema corresponden a las clases de lenguaje que pueden ser reconocidas utilizando diferentes modelos de máquinas de computación.

Tipos de gramáticas

- Una gramática de **tipo 0** no impone ninguna restricción a sus producciones.
- Una gramática de **tipo 1** puede tener producciones ele la forma $w_1 \rightarrow w_2$, donde la longitud de w_2 es mayor que la de w_1, o de la forma $w_1 \rightarrow \lambda$.
- Una gramática de **tipo 2** sólo puede tener producciones de la forma $w_1 \rightarrow w_2$, donde w_1 es un único símbolo no terminal.
- Una gramática de **tipo 3** sólo puede tener producciones de la forma $w_1 \rightarrow w_2$, con $w_1 = A$, y bien $w_2 = aB$ o bien $w_2 = a$, siendo A y B símbolos no terminales y a un símbolo terminal, o con $w_1 = S$ y $w_2 = \lambda$.

Nota

De estas definiciones se sigue que toda gramática de tipo 3 es de tipo 2, toda gramática de tipo 2 es de tipo 1 y toda gramática de tipo 1 es de tipo 0.

Las gramáticas de tipo 2 también se llaman **gramáticas libres de contexto** o **independientes de contexto**, puesto que un símbolo no terminal que esté en el lado izquierdo de una producción puede ser reemplazado en una cadena siempre que aparezca independientemente de lo que figure en la cadena.

Un lenguaje generado por una gramática de tipo 2 se llama **lenguaje libre de contexo** o **independiente del contexto**.

Cuando se tiene una producción de la forma $lw_1r \rightarrow lw_2r$ (pero no de la forma $w_1 \rightarrow w_2$), la gramática se denomina de tipo I, **sensible al contexto** o **dependiente del contexto**, puesto que w_1 puede ser reemplazado por w_2, sólo cuando está entre las cadenas l y r.

La gramáticas de tipo 3 se llaman también **regulares**. Un lenguaje generado por una gramática regular se llama **regular**. En una sección posterior se analizará la relación existente entre las gramáticas regulares y las máquinas de estado finito.

El diagrama de Venn de la Figura 36 muestra la relación que existe entre los diferentes tipos de gramáticas.

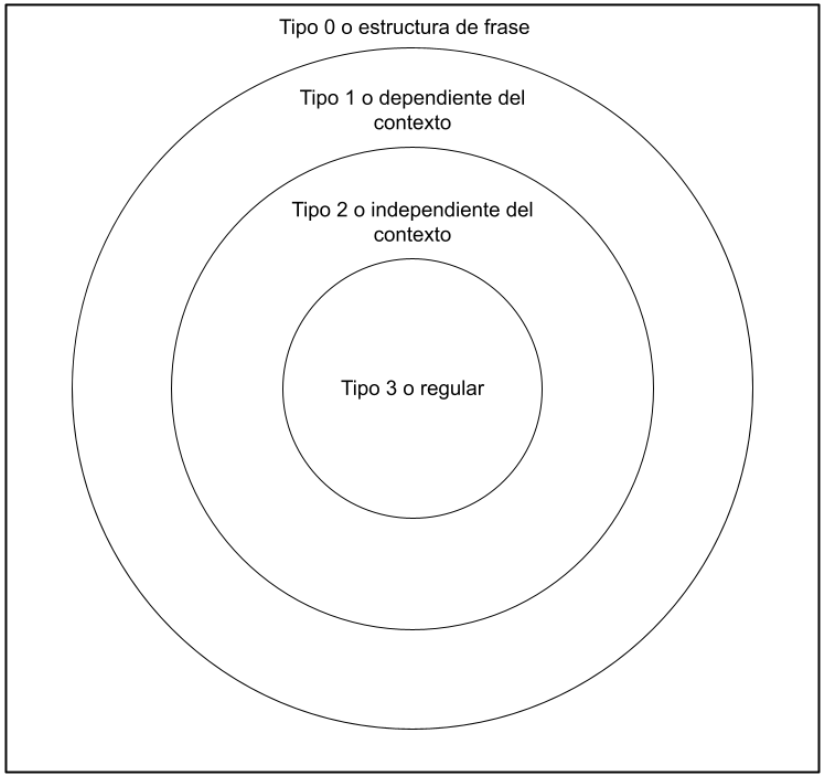

Figure: Tipos de gramáticas

Ejemplo 8

En el Ejemplo 6 vimos que $\{0^m 1^n | m, n = 0, 1, 2, \dots\}$ es un lenguaje regular, puesto que puede generarse mediante una gramática regular, a saber, la gramática G_2 del Ejemplo 6.

Las gramáticas libres de contexto y las regulares desempeñan un papel muy importante en los lenguajes de programación. Las gramáticas libres de contexto se utilizan para definir la sintaxis de casi todos los lenguajes de programación. Estas gramáticas son lo bastante potentes como para definir una amplia variedad de lenguajes. Además, se pueden diseñar algoritmos eficientes que determinen cuando y cómo generar una cadena. Las gramáticas regulares se utilizan para buscar ciertos patrones dentro de un texto y en el análisis léxico. que es el proceso en el que un analizador sintáctico toma una secuencia de entrada y la transforma en componentes léxicos (en inglés, *tokens*) .

Ejemplo 9

Del Ejemplo 5 se sigue que $\{0^n1^n|n = 0, 1, 2, \dots\}$ es un lenguaje libre de contexto, puesto que las producciones en es ta gramática son $S \to 0S1$ y $S \to \lambda$. Sin embargo, éste no es un lenguaje regular. Esta afirmación se justificará más adelante.

Ejemplo 10

El conjunto $\{0^n1^n|n = 0, 1, 2, \dots\}$ es un lenguaje dependiente del contexto, pues se puede generar mediante una gramática de tipo I. como muestra el Ejemplo 7, pero no es un lenguaje de tipo 2.

Table: Tipos de gramáticas

Tipo	Restricciones en las producciones $w_1 \rightarrow w_2$
0	Sin restricciones
1	$l(w_1) < l(w_2)$ o $w_2 = \lambda$
2	$w_1 = A$, siendo A un símbolo terminal
3	$w_1 = A$ y $w_2 = aB$ o $w_2 = a$, siendo $A \in N, B \in N$ y $a \in T$, o $S \rightarrow \lambda$

Outline

Introducción

Una derivación en el lenguaje generado por una gramática libre de contexto se puede representar gráficamente mediante un árbol con raíz ordenado, denominado **árbol de derivación**. La raíz de este árbol representa al símbolo inicial. Los nodos internos del árbol representan los símbolos no terminales que aparecen en la derivación. Las hojas del árbol representan los símbolos terminales. Si la producción $A \to w$ se utiliza en la derivación, donde w es una palabra, el vértice que representa a A tiene como hijos en el árbol a todos los vértices que representan los símbolos de w, ordenados de izquierda a derecha.

Ejemplo 11

Construye un árbol de derivación para la derivación de *el conejo hambriento come rápidamente*, dada en la introducción.

Solución

El árbol de derivación se muestra en la Figura 43

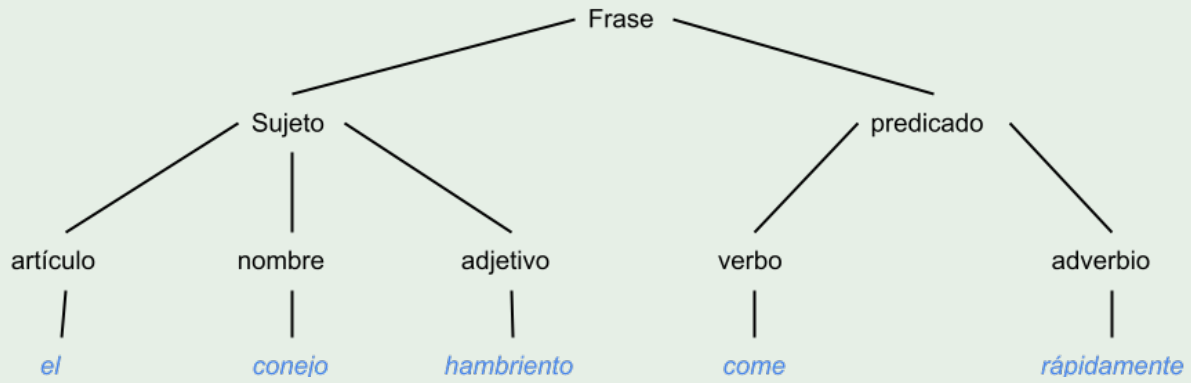

Figure: Un árbol de derivación

Ejemplo 12

Determina si la palabra *cbab* pertenece o no al lenguaje generado por la gramática $G = (V.T.S, P)$, donde $V = \{a, b, c, A, B, C, S\}$, $T = \{a, b, c\}$, S es el símbolo inicial y las producciones son

$$S \to AB$$
$$A \to Ca$$
$$B \to Ba$$
$$B \to Cb$$
$$B \to b$$
$$C \to cb$$
$$C \to b$$

Análisis descendete

Solución

Comenzamos con S y se intenta derivar *cbab* usando una serie de producciones. Puesto que sólo hay una producción con S en el lado izquierdo, debemos empezar con $S \Rightarrow AB$. Seguidamente utilizamos la única producción que tiene a A en su lado izquierdo, a saber, $A \rightarrow Ca$, para obtener $S \Rightarrow AB \Rightarrow CaB$. Puesto que *cbab* comienza con los símbolos *cb*, utilizamos la producción $C \rightarrow cb$. Ésto da lugar a $S \Rightarrow Ab \Rightarrow CaB \Rightarrow cbaB$. Finalizamos utilizando la producción $B \rightarrow b$ para obtener $S \Rightarrow AB \Rightarrow CaB \Rightarrow cbaB \Rightarrow cbab$.

El método utilizado se llama **análisis descendente**, puesto que parte del símbolo inicial y procede aplicando producciones sucesivamente.

Análisis ascendente

Solución

Puesto que *cbab* es la palabra que hay que derivar, podemos utilizar la producción $C \rightarrow cb$, obteniendo $Cab \Rightarrow cbab$. Seguidamente, podemos usar la producción $A \rightarrow Ca$ para obtener $Ab \Rightarrow Cab \Rightarrow cbab$. Utilizando la producción $B \rightarrow b$, se obtiene $AB \Rightarrow Ab \Rightarrow Cab \Rightarrow cbab$. Finalmente, mediante la producción $S \rightarrow AB$ se completa la derivación para *cbab*, que es $S \Rightarrow AB \Rightarrow Ab \Rightarrow Cab \Rightarrow cbab$

En este caso, se derivó hacia atrás, y se denomina **análisis ascendente**.

Outline

Introducción

En ocasiones se utiliza otra notación para especificar una gramática de tipo 2 llamada **forma de Backus-Naur (FBN)**, llamada así en honor a John Backus. que la inventó. y Peter Naur, Quien la modificó para utilizarla en las especificaciones del lenguaje de programación ALGOL.

La forma de Backus-Naur se emplea para especificar las reglas sintácticas de muchos lenguajes de programación, incluido el lenguaje Java.

Las producciones en una gramática de tipo 2 tienen un solo símbolo no tem1inal en su lado izquierdo. En lugar de enumerar las producciones separadamente, podemos combinar todas aquellas que tienen el mismo símbolo no terminal en el lado izquierdo.

En lugar de utilizar el símbolo $\rightarrow$ en una producción utilizaremos el símbolo $::=$. Colocaremos los símbolos no terminales entre corchetes, $\langle \cdot \rangle$, y listaremos todos los lados derechos de las producciones en una misma línea. separados por barras.

Ejemplo

Las producciones $A \rightarrow Aa$, $A \rightarrow a$ y $A \rightarrow AB$ se pueden agrupar dando lugar a

$$\langle A \rangle ::= \langle A \rangle a | a | \langle A \rangle \langle B \rangle$$

Ejemplo 13

En ALGOL 60, un identificador (que es el nombre de una entidad como, por ejemplo, una variable) consiste en una cadena de caracteres alfanuméricos (esto es, letras y número) que deben comenzar por una letra. Podemos utilizar estas reglas en la forma de Backus-Naur para describir un conjunto de identificadores válidos:

$$\langle identificador \rangle ::= \langle letra \rangle | \langle identificador \rangle \langle letra \rangle | \langle identificador \rangle \langle digito \rangle$$
$$\langle letra \rangle ::= a|b|c|\ldots|z$$
$$\langle digito \rangle ::= 0|1|2|3|4|5|6|7|8|9$$

Podemos producir el identificador válido $x99a$ utilizando la primera regla para reemplazar $\langle identificador \rangle$ por $\langle identificador \rangle \langle letra \rangle$, luego la segunda regla para obtener $\langle identificador \rangle a$, después la primera regla dos veces para obtener $\langle identificador \rangle \langle digito \rangle \langle digito \rangle a$.

Ejemplo 14

¿Cuál es la forma de Backus-Naur de la gramática para el subconjunto del español descrito en la introducción?

Solución

$$\langle \text{frase} \rangle ::= \langle \text{sujeto} \rangle \langle \text{predicado} \rangle$$
$$\langle \text{sujeto} \rangle ::= \langle \text{artículo} \rangle \langle \text{nombre} \rangle \langle \text{adjetivo} \rangle | \langle \text{artículo} \rangle \langle \text{nombre} \rangle$$
$$\langle \text{predicado} \rangle ::= \langle \text{verbo} \rangle \langle \text{adverbio} \rangle | \langle \text{verbo} \rangle$$
$$\langle \text{artículo} \rangle ::= \text{un} | \text{el}$$
$$\langle \text{adjetivo} \rangle ::= \text{grande} | \text{hambriento}$$
$$\langle \text{nombre} \rangle ::= \text{conejo} | \text{matemático}$$
$$\langle \text{verbo} \rangle ::= \text{come} | \text{salta}$$
$$\langle \text{adverbio} \rangle ::= \text{rápido} | \text{salvajemente}$$

Ejemplo 15

Halla la forma de Backus-Naur para la producción de los enteros con signo en notación decimal. (Un **entero con signo** es un nntural precedido por un signo más o un signo menos).

Solución

La forma de Backus-Nauro para una gramática que produce enteros con signo es

$$\langle \text{Entero con signo} \rangle ::= \langle \text{signo} \rangle \langle \text{entero} \rangle$$
$$\langle \text{signo} \rangle ::= +|-$$
$$\langle \text{entero} \rangle ::= \langle \text{dígito} \rangle | \langle \text{digito} \rangle \langle \text{entero} \rangle$$
$$\langle \text{dígito} \rangle ::= 0|1|2|3|4|5|6|7|8|9$$