

# Gramàtiques incontextuales y autòmats con pila

Joan Vancells i Flotats

PID\_00216112



# Índice

<b>Introducción</b>	5
<b>Objetivos</b>	6
<b>1. Conceptos introductorios</b>	7
1.1. Gramática generativa	7
1.1.1. Definición de gramática generativa	7
1.1.2. Derivación de las palabras de un lenguaje	8
1.1.3. Lenguaje generado por la gramática	9
1.1.4. Jerarquía de lenguajes de Chomsky	11
1.2. Gramática incontextual	13
1.2.1. Definición de gramática incontextual	13
1.2.2. Ejemplos de gramáticas incontextuales	13
1.3. Lenguajes incontextuales	16
<b>2. Árbol de derivación y ambigüedad</b>	17
2.1. Construcción de un árbol de derivación	18
2.2. Árboles de derivación y derivaciones	21
2.3. Lenguajes inherentemente ambiguos	25
<b>3. Verificación de gramáticas</b>	27
<b>4. Simplificación de una gramática</b>	31
4.1. Gramática limpia	31
4.2. Gramática pelada	34
<b>5. Formas normales</b>	38
5.1. Forma normal de Chomsky	38
5.2. Forma normal de Greibach	41
<b>6. Autómatas con pila</b>	47
6.1. Definición de autómata con pila	48
6.2. Descripción instantánea de un autómata con pila	49
6.3. Lenguaje aceptado por un autómata con pila	50
6.4. Ejemplos	51
6.5. Equivalencia entre autómatas con pila y gramáticas incontextuales	58
<b>7. Propiedades de los lenguajes incontextuales</b>	62
7.1. Lema del bombeo	62
7.2. Propiedades de cierre	65
7.3. Algoritmos de decisión	69

<b>Resumen .....</b>	<b>72</b>
<b>Ejercicios de autoevaluación .....</b>	<b>73</b>
<b>Solucionario .....</b>	<b>74</b>
<b>Glosario .....</b>	<b>79</b>
<b>Bibliografía .....</b>	<b>80</b>

## Introducción

En este módulo didáctico centramos nuestra atención en una clase de lenguajes que reciben el nombre de **lenguajes incontextuales** o **de contexto libre**\*, que incluyen y amplían los lenguajes regulares presentados en el módulo “Autómatas finitos y lenguajes regulares”. Del mismo modo que los regulares, los lenguajes incontextuales tienen una gran importancia práctica, sobre todo en relación con la definición y traducción de lenguajes de programación, ya que, de hecho, se pueden considerar los fundamentos teóricos en la construcción de los compiladores.

\* *Context-free*, en inglés.

El concepto de *gramática generativa* adquirirá una especial importancia en el sentido de conjunto de reglas que nos permiten producir el conjunto de palabras que forman un lenguaje, además de los conceptos ya introducidos, como *autómata finito*, *no-determinismo*, etc.

La clase de los lenguajes incontextuales es más amplia que la de los lenguajes regulares y tiene un mayor poder expresivo, a pesar de que está más limitada. Principalmente, no existe el concepto paralelo al de *autómata finito mínimo*, y esto impide tener un procedimiento general para demostrar que dos autómatas reconocen el mismo lenguaje. Como veremos, las máquinas abstractas empleadas para reconocer lenguajes incontextuales son una pequeña ampliación de los denominados *autómatas finitos*: los **autómatas con pila**.

Estudiaremos a fondo las propiedades principales de los lenguajes incontextuales, como hemos hecho con los lenguajes regulares, y también veremos sus limitaciones: hay lenguajes que no son incontextuales y que están generados por gramáticas que son menos restrictivas que las gramáticas incontextuales.

## Objetivos

En los materiales didácticos de este módulo, el estudiante encontrará las herramientas necesarias para alcanzar los siguientes objetivos:

1. Saber encontrar a partir de la definición de un lenguaje la gramática que lo genera.
2. Reconocer gramáticas ambiguas.
3. Tener una idea intuitiva de verificación de gramáticas (demostrar que efectivamente una gramática genera un determinado lenguaje).
4. Saber simplificar gramáticas.
5. Ser capaz de expresar una gramática en diferentes formas normales.
6. Saber encontrar un autómata con pila que reconozca un cierto lenguaje incontextual.
7. Demostrar que un cierto lenguaje no es incontextual.

## 1. Conceptos introductorios

### 1.1. Gramática generativa

En el módulo “Alfabetos, palabras y lenguajes” hemos definido *lenguaje* como un conjunto de palabras sobre un alfabeto determinado. Hemos visto que los lenguajes finitos se pueden describir mediante una simple enumeración, y los lenguajes infinitos, mediante plantillas\*, por medio de las propiedades que cumplen las palabras que los forman o por una mezcla de ambas. En cierta forma, también se puede considerar que un autómata finito describe el lenguaje que reconoce (formado por las palabras que acepta).

\* Como las expresiones regulares en el caso de lenguajes regulares.

En un momento determinado surgió la necesidad de disponer de herramientas más específicas para definir lenguajes. Fue Noam Chomsky quien introdujo las gramáticas formales o generativas como herramientas para modelizar la estructura gramatical de las lenguas.

Básicamente, una **gramática generativa** es un conjunto finito de reglas de producción, cuya aplicación repetida nos proporciona el conjunto de palabras de un determinado lenguaje.

Veámoslo de una manera un poco más formal.

#### 1.1.1. Definición de gramática generativa

Una gramática generativa está formada por los cuatro componentes siguientes:

- Un alfabeto  $V$  de símbolos no terminales o variables.
- Un alfabeto  $\Sigma$  (disyunto del anterior) de símbolos terminales.
- Un conjunto de pares ordenados  $P$  de reglas de producción, tal que  $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$  y el primer elemento del par contiene como mínimo un símbolo de  $V$  (y, en cambio, el segundo elemento puede ser la palabra vacía  $\lambda$ ).
- Un símbolo inicial o axioma  $S \in V$ .


Recordad que el alfabeto  $\Sigma$  de **símbolos terminales** es el conjunto de letras que acaban formando cualquier palabra del lenguaje, habitualmente  $\{a, b\}$  o  $\{0, 1\}$ , expresado en general con letras minúsculas.

Los **símbolos no terminales** o variables del alfabeto  $V$  se denotan normalmente con letras mayúsculas  $\{A, B, C, \dots, Z\}$ . De éstas, se distingue y se reserva una, la  $S$ , para indicar el **símbolo inicial** o de partida. Este símbolo se utiliza para comenzar las derivaciones de las palabras del lenguaje.

El alma de toda la gramática son las **reglas de producción**  $P$ , cada una de las cuales está formada por un par ordenado  $(Q, R)$ , y se escriben en la forma  $Q \rightarrow R$ . Así, el primer elemento del par ordenado recibirá el nombre de **parte izquierda de la regla**, y el segundo elemento del par ordenado, **parte derecha de la regla**. Estas producciones se utilizan para derivar palabras nuevas a partir de palabras dadas mediante la sustitución de un fragmento igual que la parte izquierda de una regla por la parte derecha de la misma regla.


Algunos ejemplos de gramáticas son los siguientes:

- $(\{S\}, \{a\}, P, S)$ , donde las reglas de  $P$  son  $S \rightarrow \lambda$  y  $S \rightarrow aS$ .
- $(\{S\}, \{a, b\}, P, S)$ , donde las reglas de  $P$  son  $S \rightarrow \lambda$  y  $S \rightarrow aSb$ .
- $(\{S, X, Y\}, \{a, b, c\}, P, S)$ , donde las reglas de  $P$  son  $S \rightarrow abc$ ,  $S \rightarrow aXbc$ ,  $Xb \rightarrow bX$ ,  $Xc \rightarrow Ybcc$ ,  $bY \rightarrow Yb$ ,  $aY \rightarrow aaX$  y  $aY \rightarrow aa$ .

El convenio de notación, que reserva las letras mayúsculas para representar variables o símbolos no terminales (y en especial, la  $S$  como símbolo inicial) y las letras minúsculas para representar símbolos terminales, permite que podamos expresar cualquier gramática sólo con sus reglas. Esto es lo que haremos en adelante. 

### 1.1.2. Derivación de las palabras de un lenguaje

Ahora que ya tenemos definida la gramática generativa, debemos ver cómo se relaciona con la producción del conjunto de palabras de un lenguaje. Hemos dado algunas pistas en el apartado anterior; sabemos que se trata de aplicar de una manera repetida el conjunto de reglas de producción de la gramática.

Y ¿cómo se aplica una regla de producción? Se reconoce en la palabra en curso una subpalabra que coincide con la parte izquierda de la regla de producción y se sustituye esta subpalabra por la parte derecha de la regla de producción. 

El proceso de sustituir la parte izquierda de una regla que aparece en una palabra por la parte derecha de la misma regla se llama **derivación en un solo paso**

#### Al final del proceso...

... de generación de palabras del lenguaje por medio de la aplicación de las reglas de producción deben quedar sólo símbolos del alfabeto  $\Sigma$ .

Ved la definición de *derivación* en el subapartado 1.1.2. de este módulo didáctico.



#### Observad que...

... la parte derecha de la regla puede ser la palabra vacía o cualquier combinación de variables y letras, y que en la parte izquierda de la regla siempre hay como mínimo una variable, tal como hemos indicado en la definición de gramática.



y se dice que la palabra resultante de la sustitución se deriva directamente o es derivable en un solo paso de la palabra en curso antes de la sustitución.

Más formalmente, si tenemos dos palabras  $\alpha, \beta \in (V \cup \Sigma)^*$ , diremos que  $\beta$  se deriva directamente o en un paso de  $\alpha$ , y lo expresaremos  $\alpha \Rightarrow \beta$ , si hay palabras  $\chi, \delta \in (V \cup \Sigma)^*$ , y hay una regla de producción en la gramática  $Q \rightarrow R$  tal que  $\alpha = \chi Q \delta$  y  $\beta = \chi R \delta$ . Observamos que las palabras son iguales, excepto en el hecho de que donde una tiene la parte izquierda de una regla de la gramática, la otra tiene la parte derecha de la misma regla.

Veámoslo con un ejemplo:

- La palabra  $aaaaS$  se deriva directamente de la palabra  $aaaS$  (lo notaremos  $aaaS \Rightarrow aaaaS$ ) aplicando la regla  $S \rightarrow aS$ .
- $aaaSbbb \Rightarrow aaaaSbbbb$  (aplicando la regla  $S \rightarrow aSb$ ).
- $aaaaXbbbbcccc \Rightarrow aaaabXbbbbcccc$  (aplicando la regla  $Xb \rightarrow bX$ ).
- $aaaabbbbXcccc \Rightarrow aaaabbbbYbcccc$  (aplicando la regla  $Xc \rightarrow Ybcc$ ).

Diremos que una palabra  $\beta$  se deriva (se sobreentiende que en un número cualquiera de pasos) de otra palabra  $\alpha$  si hay una secuencia finita de derivaciones directas (de un solo paso) que nos llevan de una a otra. Lo expresaremos así:  $\alpha \xRightarrow{*} \beta$ .

Veamos algunos ejemplos:

- La palabra  $aaaaaaaS$  se deriva directamente de la palabra  $aaaS$  aplicando la regla  $S \rightarrow aS$  cuatro veces consecutivas. Lo notaremos  $aaaS \xRightarrow{*} aaaaaaS$  o bien  $aaaS \xRightarrow{4} aaaaaaS$ .
- $aaaSbbb \xRightarrow{*} aaaaaaSbbbbbb$  (aplicando la regla  $S \rightarrow aSb$  tres veces).
- $aaaaXbbbbcccc \xRightarrow{*} aaaabbbbXcccc$  (aplicando la regla  $Xb \rightarrow bX$  cuatro veces).
- $aaaabbbbXcccc \xRightarrow{*} aaaaYbbbbbbcccc$  (aplicando la regla  $Xc \rightarrow Ybcc$  una vez y la regla  $bY \rightarrow Yb$  cuatro veces a continuación).

En el caso de que sepamos exactamente el número de derivaciones directas, ( $n$ ) se puede poner en lugar del  $*$ .

### 1.1.3. Lenguaje generado por la gramática

Es probable que intuitivamente ya os deis cuenta de la manera en que debe salir el conjunto de palabras del lenguaje a partir de estas reglas. Ahora sólo nos falta establecer cuándo y cómo empieza y acaba la derivación. Aquí entra en juego el símbolo inicial  $S$ .

Las palabras del lenguaje generado por la gramática son todas aquellas que se pueden producir a partir de la variable  $S$ , aplicando repetidamente las diferentes reglas de la gramática en todos los órdenes posibles.

El **lenguaje generado por una gramática**  $G$  está formado por todas las palabras que sólo contienen símbolos terminales (del alfabeto  $\Sigma$ ), que se pueden derivar a partir del símbolo inicial  $S$ .

Más formalmente, lo notaremos así:

$$L(G) = \{w \in \Sigma^* \mid S \xRightarrow{*} w\}.$$

Por tanto, los símbolos no terminales sólo se utilizan como símbolos auxiliares para realizar las derivaciones. Y una derivación se acaba cuando ya no quedan más símbolos no terminales o variables en la palabra. Debe tenerse en cuenta que, si en la palabra quedan símbolos no terminales pero no se puede aplicar ninguna regla de producción, la derivación aborta.

Pasemos a ver cuáles son los lenguajes generados por las gramáticas que hemos visto en los subapartados anteriores:

1)  $S \rightarrow \lambda$  y  $S \rightarrow aS$ .

Esta gramática es la más simple. Aprovechamos para estudiarla más a fondo, generando una por una todas las palabras del lenguaje:

- $\lambda$  se deriva en un solo paso aplicando la primera regla.
- $a$  se deriva aplicando una vez la segunda regla y después la primera.
- $aa$  se deriva aplicando dos veces la segunda regla y después la primera.
- $a^n$ , en general, aplicando  $n$  veces la segunda regla y al final la primera.

Este lenguaje ya lo conocemos, es un lenguaje regular, y su gramática se llama **gramática regular**.

Dado que partiendo de  $S$  sólo podemos aplicar la segunda regla, para que no desaparezcan los símbolos no terminales y para acabar la derivación sólo podemos aplicar la primera. Entonces, queda claro que sólo podemos generar las palabras formadas únicamente por  $a$ . Más formalmente,  $L = \{a^n \mid n \geq 0\}$ .

2)  $S \rightarrow \lambda$  y  $S \rightarrow aSb$ .

Esta gramática es parecida a la anterior en cuanto a condiciones de aplicación (sólo se puede aplicar la segunda regla tantas veces como se quiera para acabar aplicando la primera una sola vez). Veamos la derivación de una palabra cualquiera:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbbb.$$

También es fácil concluir cuál es el lenguaje que genera:  $L = \{a^n b^n \mid n \geq 0\}$ .

#### Recordad que...

... según la definición, la parte izquierda de una regla de producción debe contener como mínimo un símbolo no terminal. Por tanto, si no hay símbolos no terminales no se puede aplicar ninguna regla.

Este lenguaje también lo conocemos, precisamente porque es el típico ejemplo de lenguaje que no es regular (por simple que sea y parecido al anterior).

Consultad el lema del bombeo en el apartado 6 del módulo "Autómatas finitos y lenguajes regulares" de esta asignatura.

3)  $S \rightarrow abc, S \rightarrow aXbc, Xb \rightarrow bX, Xc \rightarrow Ybcc, bY \rightarrow Yb, aY \rightarrow aaX \text{ y } aY \rightarrow aa$

Esta gramática ya es más compleja: para una misma parte izquierda ( $S$ ) se pueden aplicar dos reglas diferentes; hay reglas que tienen en la parte izquierda símbolos terminales, en cuyo caso no es tan directo ver qué se hace (a pesar de que en los ejemplos de derivación ya lo hemos empezado a hacer). Analicémosla más a fondo:

a) Aplicando sólo la primera regla, obtenemos la primera palabra del lenguaje:  $abc$ .

b) Aplicando la segunda regla, introducimos una  $X$  seguida de  $b$  que sólo podemos quitar con la tercera regla, y produce una  $X$  seguida de  $c$  que sólo se puede eliminar con la cuarta regla. Se procede así sucesivamente hasta poder decidir si la  $a$  seguida de  $Y$  la cambiamos por dos  $a$  (la séptima regla) y acabamos:

$$S \Rightarrow aXbc \Rightarrow abXc \Rightarrow abYbcc \Rightarrow aYbbcc \Rightarrow aabbcc,$$


o bien la cambiamos por dos  $a$  seguidas de  $X$  (la sexta regla) y debemos iniciar de nuevo el proceso:

$$S \Rightarrow aXbc \Rightarrow abXc \Rightarrow abYbcc \Rightarrow aYbbcc \Rightarrow aaXbbcc \Rightarrow aabXbcc \Rightarrow aabbXcc \Rightarrow aabbYbcc \Rightarrow aaYbbbcc \Rightarrow aaYbbbcc \Rightarrow aaabbcc.$$

Si comprobáis las siguientes palabras que genera la gramática, concluiréis que el lenguaje generado por la gramática es  $L = \{a^n b^n c^n \mid n > 0\}$ .

#### 1.1.4. Jerarquía de lenguajes de Chomsky

Noam Chomsky, introductor de las gramáticas generativas, también propuso una clasificación de estas gramáticas en función de la forma de sus reglas de producción. Esta clasificación establece la base de la teoría de lenguajes formales, de la que es fundador: cada tipo de gramática caracteriza un tipo diferente de lenguajes generados por éstas, y, además, cada uno está relacionado con un tipo diferente de máquina abstracta que lo reconoce.

Se distinguen cuatro **tipos de gramáticas generativas**: 

a) **Tipo 0: gramáticas generales** o de estructura de frase. Son aquellas gramáticas que no tienen ningún tipo de restricción. Su definición coincide con la de la gramática generativa: tanto en la parte derecha como en la parte izquierda de las reglas de producción puede haber cualquier combinación de letras de

los dos alfabetos, de símbolos terminales y no terminales (que tienen, como mínimo, una variable en la parte izquierda).

**b) Tipo 1: gramáticas contextuales** o sensibles al contexto. Cada regla de producción tiene la forma  $\alpha A \beta \rightarrow \alpha \chi \beta$  con  $\alpha, \beta, \chi \in (V \cup \Sigma)^*$ ,  $A \in V$  y  $\chi \neq \lambda$ , excepto para la regla  $S \rightarrow \lambda$ , y entonces  $S$  no puede aparecer en la parte derecha de las reglas.

**c) Tipo 2: gramáticas incontextuales** o de contexto libre. Cada regla de producción tiene la forma  $A \rightarrow \alpha$  con  $\alpha \in (V \cup \Sigma)^*$  y  $A \in V$  y, por tanto, la sustitución de la variable  $A$  por la palabra  $\alpha$  puede hacerse en cualquier contexto.

**d) Tipo 3: gramáticas regulares.** Cada regla de producción tiene la forma  $A \rightarrow \alpha B$  o bien  $A \rightarrow \alpha$  con  $\alpha \in \Sigma^*$  y  $A, B \in V$ .

Ya hemos comentado en el primer ejemplo de gramática 3 que el lenguaje generado era regular. Las gramáticas de tipo 3, por tanto, generan lenguajes como los ya estudiados en otro módulo de esta asignatura. Y ya sabemos que estos lenguajes regulares son reconocidos por un autómata finito.

#### Observad que...

... en el caso de las gramáticas contextuales, para aplicar la regla (sustituir el símbolo no terminal  $A$  por la palabra  $\chi$ ), el símbolo no terminal debe tener un contexto determinado (un conjunto de elementos delante y detrás). De ahí viene la denominación *contextual*.

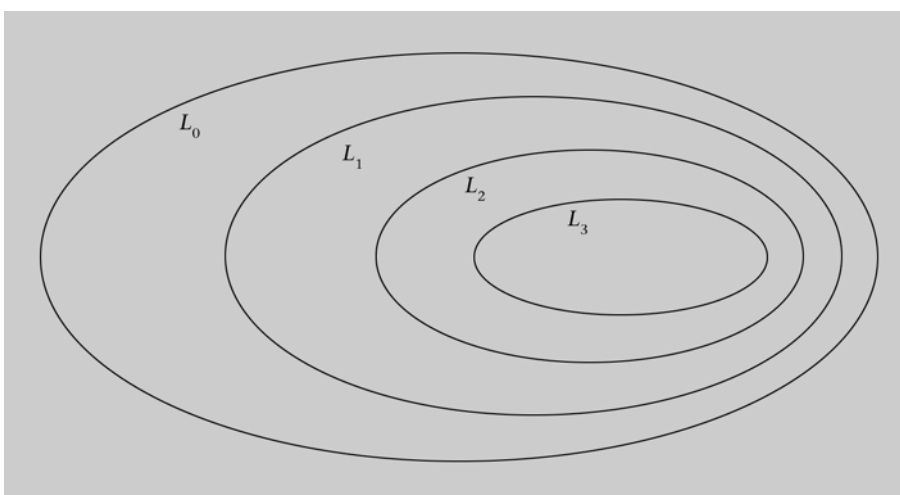
Repasad los lenguajes regulares en el subapartado 1.4. del módulo "Autómatas finitos y lenguajes regulares" de esta asignatura.

## Actividad

1.1. Para familiarizaros con las gramáticas generativas puede ser interesante que encontréis las gramáticas regulares (con las restricciones del tipo 3) para los lenguajes que hemos visto en el módulo "Autómatas finitos y lenguajes regulares".

La propiedad más interesante de esta jerarquía de lenguajes (respecto al tipo de gramática que los genera) es que cada uno está estrictamente incluido en la clase anterior. Los lenguajes regulares son un subconjunto de lenguajes incontextuales\*; los incontextuales, a su vez, son un subconjunto de los contextuales y, finalmente, los contextuales son un subconjunto de los lenguajes generales.

\* La misma definición de las gramáticas de tipo 2 y 3 nos lo corrobora: es la misma forma pero más restringida en el tipo 3.



#### Notación

- $L_0$  son los lenguajes generales (gramáticas sin restricciones).
- $L_1$  son los lenguajes contextuales.
- $L_2$  son los lenguajes incontextuales.
- $L_3$  son los lenguajes regulares (los únicos que ya hemos visto).

## 1.2. Gramática incontextual

El objeto de estudio de este módulo son los lenguajes incontextuales, es decir, los generados por las gramáticas de tipo 2 o incontextuales, y principalmente aquellos que no son regulares. Los lenguajes regulares son aquellos de que ya hemos tratado con todo detalle, y es preciso avanzar que tienen más propiedades interesantes que los incontextuales.


### 1.2.1. Definición de gramática incontextual

El concepto *gramática incontextual* ya se ha definido anteriormente, porque es una gramática generativa de tipo 2 en la jerarquía de Chomsky. Agrupémoslo todo en una única definición.

Una gramática incontextual está formada por los cuatro componentes siguientes:

- 1) Un alfabeto  $V$  de símbolos no terminales o variables.
- 2) Un alfabeto  $\Sigma$  (disjunto del anterior) de símbolos terminales.
- 3) Un conjunto de pares ordenados  $P$  de reglas de producción tal que  $P \subseteq V \times (V \cup \Sigma)^*$ . Dicho de otro modo, las reglas de producción tienen la forma  $A \rightarrow \alpha$  con  $\alpha \in (V \cup \Sigma)^*$  y  $A \in V$ .
- 4) Un símbolo inicial o axioma  $S \in V$ .

Debemos recordar que, por defecto, representamos las variables con letras mayúsculas y los símbolos terminales con letras minúsculas, como se ha hecho hasta ahora. Así, podemos expresar la gramática especificando sólo las reglas. En el caso de que aparezcan algunos caracteres, escribiremos la gramática entera.

Además, para abreviar la escritura de las reglas de producción, se suelen agrupar en una misma línea todas las que comparten la misma variable en la parte izquierda, y así sólo es necesario escribir cada variable una sola vez. Las partes que aparecen a la derecha de las reglas se escriben a la derecha, separadas por una barra vertical (|). 

### 1.2.2. Ejemplos de gramáticas incontextuales

En este subapartado estudiaremos algunos ejemplos de gramáticas incontextuales, que nos ayudarán a entender este concepto.



Así pues, vemos cómo se generaría una expresión correcta:

$$\begin{aligned}
 S &\Rightarrow (S+S) \Rightarrow (S+(S^*S)) \Rightarrow (S+(S^*(S+S))) \Rightarrow (N+(S^*(S+S))) \Rightarrow (DN+(S^*(S+S))) \Rightarrow \\
 &\Rightarrow (DD+(S^*(S+S))) \Rightarrow (1D+(S^*(S+S))) \Rightarrow (14+(S^*(S+S))) \Rightarrow (14+(N^*(S+S))) \Rightarrow \\
 &\Rightarrow (14+(D^*(S+S))) \Rightarrow (14+(7^*(S+S))) \Rightarrow (14+(7^*(N+S))) \Rightarrow (14+(7^*(D+S))) \Rightarrow \\
 &\Rightarrow (14+(7^*(4+S))) \Rightarrow (14+(7^*(4+N))) \Rightarrow (14+(7^*(4+DN))) \Rightarrow \\
 &\Rightarrow (14+(7^*(4+DDN))) \Rightarrow (14+(7^*(4+DDD))) \Rightarrow (14+(7^*(4+3DD))) \Rightarrow \\
 &\Rightarrow (14+(7^*(4+34D))) \Rightarrow (14+(7^*(4+347))).
 \end{aligned}$$

**Ejemplo 5.** De la misma manera, podemos dar una gramática que genera el conjunto de fórmulas bien formadas de la lógica de proposiciones. Tenemos  $\Sigma = \{p_1, \dots, p_n, (, ), \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ ,  $V = \{S\}$  y las reglas:

$$S \rightarrow p_i \mid \neg(S) \mid (S) \wedge (S) \mid (S) \vee (S) \mid (S) \rightarrow (S) \mid (S) \leftrightarrow (S)$$


**Ejemplo 6.** Otro ejemplo clásico de gramática incontextual es el del conjunto de palíndromos (palabras que se leen igual del derecho que del revés, de izquierda a derecha que de derecha a izquierda):

$$S \rightarrow \lambda \mid a \mid b \mid aSa \mid bSb.$$

Las dos últimas reglas generan la misma letra al principio que al final, y esto de fuera hacia dentro de la palabra. La segunda y la tercera reglas generan palabras palíndromas de longitud impar con una  $a$  y una  $b$ , respectivamente, de símbolo central. Y la palabra vacía nos permite generar los palíndromos pares.

Generamos un palíndromo aplicando las reglas en un cierto orden:

$$\begin{aligned}
 S &\Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabaSabaa \Rightarrow \\
 &\Rightarrow aababSbabaa \Rightarrow aababaSababaa \Rightarrow \\
 &\Rightarrow aabababSbababaa \Rightarrow aabababbSbbababaa \Rightarrow \\
 &\Rightarrow aabababbaSabbababaa \Rightarrow aabababbbaSaabbababaa \Rightarrow \\
 &\Rightarrow aabababbbaaaaabbababaa.
 \end{aligned}$$

Debéis haber observado que todos los ejemplos de gramáticas que hemos dado hasta ahora pueden verse como programas recursivos (son, de hecho, recurrencias). En este caso, las tres primeras reglas serían bases de la recurrencia (cada una con su función, como ya hemos indicado), y las dos últimas reglas, llamadas recursivas. 

**Ejemplo 7.** Para acabar damos una gramática incontextual que guarda una cierta relación con la del primer ejemplo de esta clasificación:

$$\begin{aligned}
 S &\rightarrow aBS \mid bAS \mid \lambda, \\
 A &\rightarrow bAA \mid a, \\
 B &\rightarrow aBB \mid b.
 \end{aligned}$$

#### Ejemplo

“Dábale arroz a la zorra el abad.”

Recordad el módulo “Recursividad” de la asignatura *Fundamentos de programación II*.



¿Sabríais decir cuál es esta relación?

Y un ejemplo de palabra generada por ésta sería el siguiente:

$$S \Rightarrow bAS \Rightarrow bAaBS \Rightarrow bAaB \Rightarrow bbAAaB \Rightarrow bbaAaB \Rightarrow bbaaaB \Rightarrow bbaaab.$$

### 1.3. Lenguajes incontextuales

Los lenguajes incontextuales son los generados por una gramática incontextual, tal como la hemos definido y, si  $G$  es la gramática, su lenguaje lo notaremos como  $L(G)$ . Además, está formado por el conjunto de palabras para las que hay una derivación que lleva del símbolo inicial  $S$  a la palabra considerada. Recordemos que:

$$L(G) = \{w \in \Sigma^* \mid S \xRightarrow{*} w\}.$$

Un lenguaje se denomina **lenguaje incontextual** cuando hay una gramática incontextual que lo genera.

Los lenguajes generados por las gramáticas del subapartado anterior son ejemplos de gramáticas incontextuales.

- $a^n b^n$ .
- El lenguaje de las cadenas de paréntesis balanceados.
- El lenguaje de las cadenas de paréntesis (de dos tipos) balanceados.
- El lenguaje de las expresiones aritméticas bien definidas.
- El lenguaje de las fórmulas bien formadas de la lógica proposicional.
- El lenguaje de los palíndromos.
- El lenguaje de todas las palabras que tienen tantas  $a$  como  $b$ .

Pueden existir varias formas de expresar una gramática para que genere un determinado lenguaje; más adelante veremos cuál es el formato que más nos interesa.

Dos gramáticas son **gramáticas equivalentes** cuando generan el mismo lenguaje. Formalmente,  $G_1$  es equivalente a  $G_2$  si  $L(G_1) = L(G_2)$ .



## 2. Árbol de derivación y ambigüedad

Para mostrar que una palabra está generada por una gramática (y que, por tanto, pertenece al lenguaje generado por la gramática), damos su derivación paso a paso. Una manera alternativa de hacerlo, quizá más clara y que nos será útil para otras cosas, es dar su árbol de derivación.

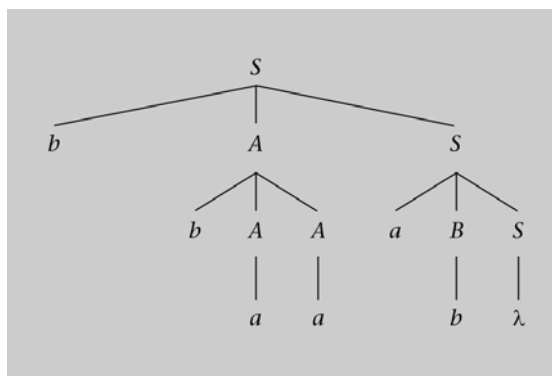
Dada una gramática y un árbol tal que todos sus nodos están etiquetados con símbolos terminales o variables o  $\lambda$ , diremos que es un **árbol de derivación** (o **sintáctico**) **respecto a la gramática** si para todo nodo interior etiquetado con una variable  $A$  y sus hijos con etiquetas  $X_1, X_2, \dots, X_n$  tenemos que  $A \rightarrow X_1X_2 \dots X_n$  es una regla de producción de la gramática.

Veamos un ejemplo de árbol de derivación con respecto a la gramática incontextual:

$$\begin{aligned} S &\rightarrow aBS \mid bAS \mid \lambda, \\ A &\rightarrow bAA \mid a, \\ B &\rightarrow aBB \mid b, \end{aligned}$$

que corresponde a la derivación siguiente de una palabra de la gramática:

$$S \Rightarrow bAS \Rightarrow bAaBS \Rightarrow bAaB \Rightarrow bbAAaB \Rightarrow bbaAaB \Rightarrow bbaaaB \Rightarrow bbaaab.$$



### Observad que...

... si leéis las hojas del árbol de derivación de izquierda a derecha, obtenéis la palabra derivada *bbaaab*.

Como ya indica su nombre, el árbol es diferente para cada derivación, pero no necesariamente para cada palabra: puede ser que una misma palabra tenga más de un árbol de derivación.

Las gramáticas en las que es posible construir dos árboles de derivación diferentes para una misma palabra se denominan **gramáticas ambiguas**. En caso contrario, hablaremos de **gramáticas inambiguas** o de **gramáticas no ambiguas**.

## 2.1. Construcción de un árbol de derivación

A continuación, damos una definición más formal de árbol de derivación, especificando claramente cómo se debe construir.

Un árbol es un **árbol de derivación** con respecto de una gramática  $G$  con componentes  $(\Sigma, V, P, S)$  si se cumplen las siguientes condiciones:

- 1) Cada nodo tiene una etiqueta, que es un símbolo de  $V \cup \Sigma \cup \lambda$ .
- 2) La etiqueta de la raíz del árbol es la variable  $S$ .
- 3) Si un nodo es interno y tiene la etiqueta  $A$ ,  $A$  debe estar en  $V$ .
- 4) Si un nodo interno tiene la etiqueta  $A$  y sus hijos, de izquierda a derecha, tienen etiquetas  $X_1, X_2, \dots, X_n$ , respectivamente, entonces  $A \rightarrow X_1 X_2 \dots X_n$  debe ser una regla de producción en  $P$ .
- 5) Si un nodo tiene etiqueta  $\lambda$ , entonces es una hoja y es el único hijo de su padre.

Ahora construiremos los árboles de derivación de las palabras que dábamos como ejemplo de derivación de las diferentes gramáticas incontextuales.

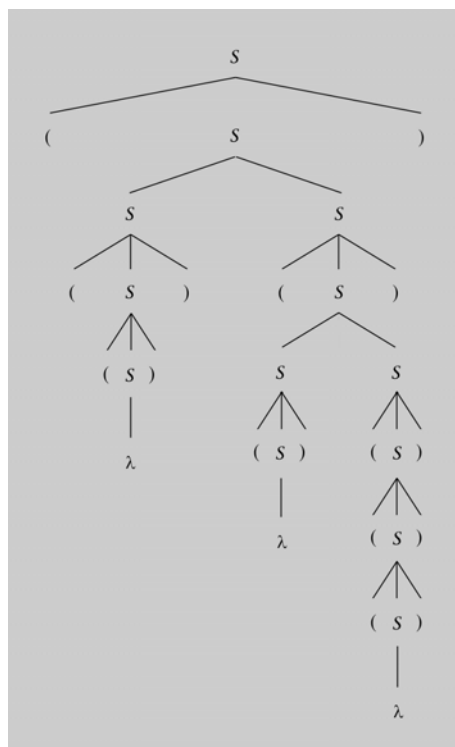
Recordad que hemos visto algunos ejemplos de gramáticas incontextuales en el subapartado 1.2.2. de este módulo didáctico.

Tened en cuenta que para cada paso de derivación tenemos el desdoblamiento de una rama del árbol formada por la regla aplicada (raíz = parte izquierda, hijos = parte derecha):

- 1) Para  $S \rightarrow \lambda \mid (S) \mid SS$  con la siguiente derivación:

$$\begin{aligned} S &\Rightarrow (S) \Rightarrow (SS) \Rightarrow ((S)S) \Rightarrow (((S))S) \Rightarrow ((())S) \Rightarrow ((())(S)) \Rightarrow ((())(SS)) \Rightarrow \\ &\Rightarrow ((())((S)S)) \Rightarrow ((())(())S) \Rightarrow ((())(())((S))) \Rightarrow ((())(())((S))) \Rightarrow \\ &\Rightarrow ((())(())((S)))) \Rightarrow ((())(())((S))))). \end{aligned}$$

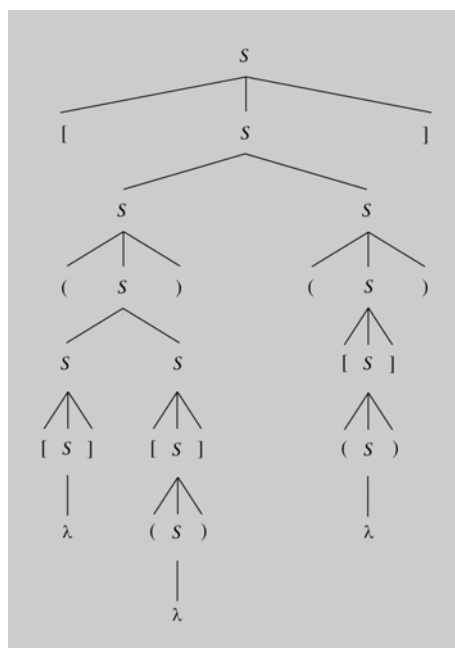
El árbol de derivación que obtenemos es el que podéis ver a continuación:



2) En el caso de la gramática  $S \rightarrow \lambda \mid (S) \mid [S] \mid SS$ , su derivación es:

$$\begin{aligned}
 S &\Rightarrow [S] \Rightarrow [SS] \Rightarrow [(S)S] \Rightarrow [(SS)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \\
 &\Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S] \Rightarrow [(S)S)S]
 \end{aligned}$$

Así, el árbol de derivación que conseguimos es el siguiente:



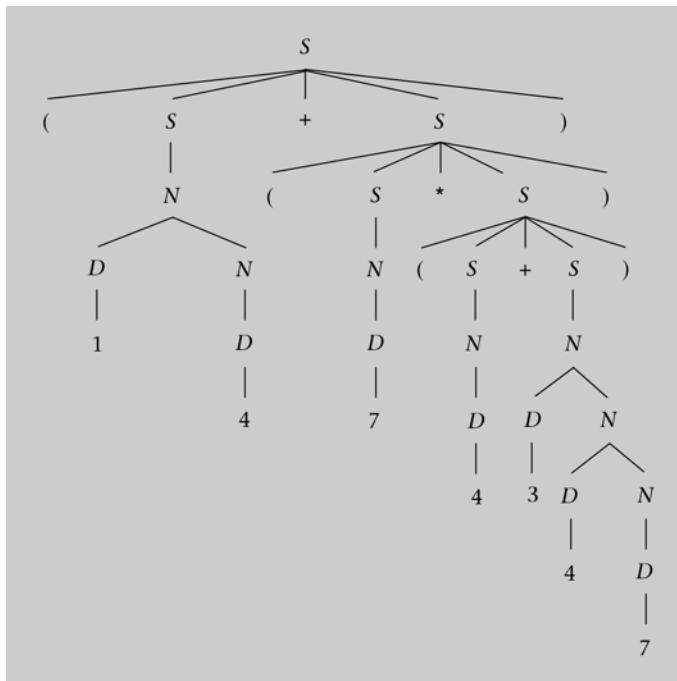
3) Vemos ahora el caso de  $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (, ), +, *\}$ ,  $V = \{S, D, N\}$ , y las reglas:

$$\begin{aligned} S &\rightarrow N \mid (S+S) \mid (S^*S), \\ N &\rightarrow D \mid DN, \\ D &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9. \end{aligned}$$

Recordad que su derivación es la siguiente:

$$\begin{aligned} S &\Rightarrow (S+S) \Rightarrow (S+(S^*S)) \Rightarrow (S+(S^*(S+S))) \Rightarrow (N+(S^*(S+S))) \Rightarrow (DN+(S^*(S+S))) \Rightarrow \\ &\Rightarrow (DD+(S^*(S+S))) \Rightarrow (1D+(S^*(S+S))) \Rightarrow (14+(S^*(S+S))) \Rightarrow (14+(N^*(S+S))) \Rightarrow \\ &\Rightarrow (14+(D^*(S+S))) \Rightarrow (14+(7^*(S+S))) \Rightarrow (14+(7^*(N+S))) \Rightarrow \\ &\Rightarrow (14+(7^*(D+S))) \Rightarrow (14+(7^*(4+S))) \Rightarrow (14+(7^*(4+N))) \Rightarrow \\ &\Rightarrow (14+(7^*(4+DN))) \Rightarrow (14+(7^*(4+DDN))) \Rightarrow (14+(7^*(4+DDD))) \Rightarrow \\ &\Rightarrow (14+(7^*(4+3DD))) \Rightarrow (14+(7^*(4+34D))) \Rightarrow (14+(7^*(4+347))). \end{aligned}$$

Y, por tanto, el árbol de derivación correspondiente es el que presentamos a continuación:



4) Para la gramática  $S \rightarrow \lambda \mid a \mid b \mid aSa \mid bSb$ , hemos llegado a la siguiente derivación:

$$\begin{aligned} S &\Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabaSabaa \Rightarrow aababSbabaa \Rightarrow \\ &\Rightarrow aababaSababaa \Rightarrow aabababSbababaa \Rightarrow aabababbSbbababaa \Rightarrow \\ &\Rightarrow aabababbaSabbababaa \Rightarrow aabababbbaSaabbababaa \Rightarrow \\ &\Rightarrow aabababbbaaaaabbababaa. \end{aligned}$$

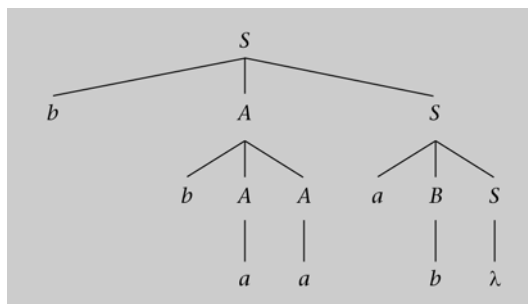


derivaciones de una misma palabra pueden quedar representadas en un mismo árbol de derivación. Nosotros hemos hecho lo mismo, pero en un orden diferente.

En el ejemplo del inicio de este apartado hemos obtenido una derivación para la palabra *bbaaab*. Ahora podemos dar la siguiente derivación para la misma palabra *bbaaab*:

$$S \Rightarrow bAS \Rightarrow bbAAS \Rightarrow bbaAS \Rightarrow bbaaS \Rightarrow bbaaaBS \Rightarrow bbaaabS \Rightarrow bbaaab.$$

Así, el árbol de derivación resultante sería el siguiente:



Ahora podéis observar que el árbol de derivación es el mismo para las dos derivaciones.

## Actividad

2.1. Hallad todas las derivaciones que lleven a la misma palabra, *bbaaab*, y que compartan el mismo árbol de derivación.

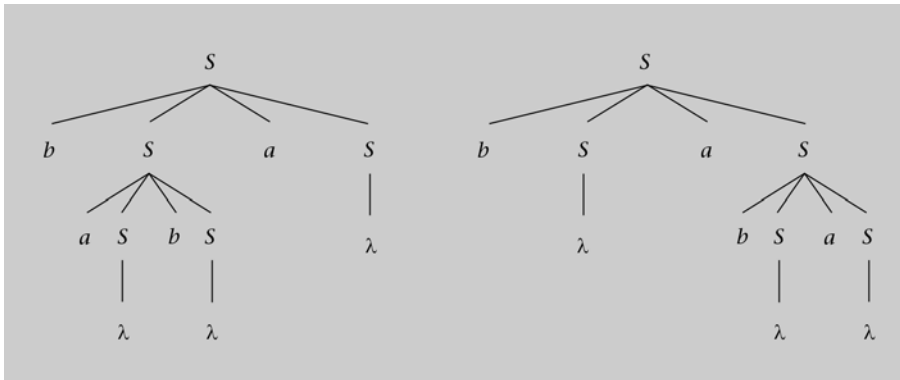
Como hemos podido observar, todas las derivaciones que corresponden a un mismo árbol de derivación son equivalentes.

De aquí viene el hecho de definir *gramática ambigua* como aquella en la que es posible construir dos árboles de derivación diferentes que corresponden a la misma palabra, y no dos derivaciones diferentes de la misma palabra. Así, eliminamos las derivaciones que se consideran equivalentes.

Observad que, mientras haya una sola palabra con dos árboles de derivación diferentes, la gramática ya puede considerarse ambigua. Por tanto, para demostrar que una gramática es inambigua, se debe demostrar que no puede haber ninguna palabra en la gramática que tenga dos árboles de derivación.

Para demostrar que una gramática es ambigua, sólo es necesario encontrar una palabra concreta para la que se puedan construir dos árboles de derivación diferentes. Se trata, por tanto, de buscar esta palabra.

Por ejemplo, la gramática  $S \rightarrow aSbS \mid bSaS \mid \lambda$  que genera el lenguaje de todas las palabras que tienen tantas  $a$  como  $b$ ,  $L = \{w \mid |w|_a = |w|_b\}$ , es una gramática ambigua, es decir, hay dos árboles de derivación diferentes para la palabra *baba* (también para otras palabras), que son los siguientes:



Para demostrar que una gramática es inambigua, es necesario demostrar que dos derivaciones diferentes (no equivalentes) deben llevar necesariamente a dos palabras diferentes, es decir, que no pueden producir la misma palabra.

Esto se puede apreciar intuitivamente observando la gramática y haciendo un seguimiento, o bien más formalmente mediante una reducción al absurdo\*. En cierto modo, si intentamos encontrar una palabra con dos árboles de derivación diferentes, llegamos a la conclusión de que es imposible, y que, si en un momento determinado de la derivación aplicamos una regla u otra sobre una misma variable, accedemos a generar conjuntos de palabras disyuntos (las palabras posibles por un lado no coinciden con las palabras posibles por el otro).

\* Es decir, suponiendo que sí que es ambigua y llegando a una contradicción.

Recuperemos la otra gramática incontextual que genera el lenguaje de todas las palabras que tienen tantas  $a$  como  $b$ ,  $L = \{w \mid |w|_a = |w|_b\}$ :

$$\begin{aligned} S &\rightarrow aBS \mid bAS \mid \lambda, \\ A &\rightarrow bAA \mid a, \\ B &\rightarrow aBB \mid b. \end{aligned}$$

Esta gramática es inambigua. Intuitivamente, si partimos de  $S$  y aplicamos la primera regla, hacemos aparecer una  $a$ ; si aplicamos la segunda, una  $b$ , y, si aplicamos la tercera, desaparece la  $S$ . Por tanto, por muchas cosas que hagamos después, las palabras ya serán diferentes (comienzan diferente). Pasa lo mismo para las cuatro reglas restantes.

**Observad que...**

... para generar un mismo lenguaje puede haber una gramática ambigua y una inambigua que sean equivalentes.

Para demostrar que una gramática es inambigua, también puede ser útil intentar construir los dos árboles de derivación en paralelo y tratar sucesivamente de aplicar las diferentes reglas alternativas para una misma variable, descartando los casos en los que quede claro que ya no podemos derivar la misma palabra (una especie de reducción al absurdo constructiva).

En el último ejemplo que hemos visto, sería algo así (la raíz es siempre  $S$ ):

- si aplicamos la primera regla, generamos una palabra que empieza por  $a$ ;
- si aplicamos la segunda, una palabra que comienza por  $b$ ;
- si aplicamos la tercera, la palabra vacía.


Dado que se trata de conjuntos disjuntos, llegamos a la conclusión de que los dos árboles de derivación deben comenzar con la misma regla para poder derivar la misma palabra. Supongamos que tomamos la primera regla:



Aplicamos el mismo razonamiento a la variable  $B$  (aplicar la primera o la segunda regla de  $B \rightarrow aBB \mid b$ ):

- si aplicamos la primera ( $B \rightarrow aBB$ ), tendremos palabras que comienzan por  $aa$ ;
- si aplicamos la segunda ( $B \rightarrow b$ ), palabras que empiezan por  $ab$ .

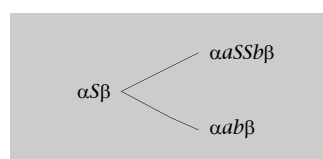
De nuevo hemos obtenido conjuntos excluyentes. Por tanto, para llegar a la misma palabra debemos poner la misma regla (y, por tanto, el mismo árbol). De esta forma, sucesivamente probaremos todas las combinaciones.

El proceso intuitivo-constructivo anterior se puede formalizar un poco más generalizándolo para cada variable de la gramática. Se trata, considerando cada variable en un cierto contexto, de aplicar las diferentes reglas alternativas para aquella variable y después desarrollarlas por cada lado hasta que obtengamos las diferencias. 

Veámoslo con un ejemplo:

$$S \rightarrow aSSb \mid ab.$$

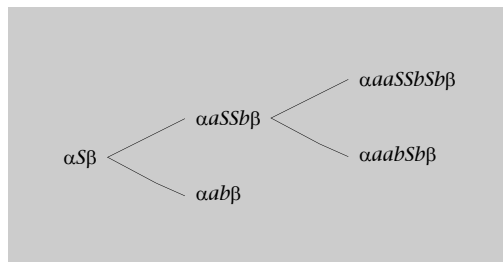
Esta gramática no es ambigua. Demostrémoslo:



Con un paso de derivación, aplicando reglas diferentes, aún no podemos concluir que no se pueda llegar a la misma palabra (las dos comienzan por  $\alpha a$ , y



falta ver cómo evoluciona la primera  $S$  en la siguiente derivación). Aplicamos otro paso a la palabra superior (de nuevo, todas las alternativas):



Ahora ya queda claro que no puede volver a salir la misma palabra: en la parte superior aparecen dos  $a$ , y en la parte inferior,  $ab$ . En un cierto momento de la derivación, si se aplica la primera o la segunda regla, ya no se puede generar la misma palabra. Por tanto, la gramática es inambigua.

Es preciso seguir esta metodología para cada variable de la gramática. Si no salen las diferencias, será señal de que nos hemos equivocado y que realmente era ambigua. El mismo desarrollo que hemos realizado nos dará una pista clara de una palabra que desenmascare la ambigüedad. Sólo es necesario darla con sus dos árboles y tendremos el trabajo terminado.

### 2.3. Lenguajes inherentemente ambiguos

Acabamos de ver que hay gramáticas ambiguas y gramáticas no ambiguas. También hemos visto un ejemplo de lenguaje que tiene como mínimo dos gramáticas equivalentes que lo generan, una de las cuales es ambigua, y la otra, no ambigua. Como regla general, nos interesará, siempre que sea posible, tener gramáticas inambiguas (en las que cada palabra se pueda generar de una sola manera). Sin embargo, es necesario tener en cuenta que no existe un procedimiento general (aplicable a todo lenguaje o gramática) para encontrar gramáticas inambiguas. Para evitarse problemas, es importante saber que no siempre hay una gramática inambigua para un lenguaje incontextual.

Un **lenguaje inherentemente ambiguo** es aquél que sólo está generado por gramáticas ambiguas.

Veamos algunos ejemplos de lenguajes inherentemente ambiguos:

1) El lenguaje incontextual:

$$L = \{a^n b^n c^m d^m \mid n, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n, m \geq 1\}$$

es inherentemente ambiguo.

## 2) El lenguaje incontextual:

$$L = \{a^i b^j c^k \mid i = j \vee j = k\}$$

también es inherentemente ambiguo.

Las demostraciones de que un lenguaje es inherentemente ambiguo son bastante pesadas y no aportan nada nuevo. Sólo nos interesa que quede claro que no siempre es factible encontrar una gramática inambigua.

Además, no existe un método absolutamente general para reconocer si un lenguaje es inherentemente ambiguo. Por tanto, es necesario probarlo de manera diferente en cada caso. Los problemas que no aceptan un método general de resolución se denominan **problemas indecidibles**.

### 3. Verificación de gramáticas

Hasta ahora hemos introducido una serie de gramáticas incontextuales, hemos dado algunos ejemplos de palabras derivadas de estas gramáticas y hemos indicado cuál era el lenguaje generado por cada gramática. Sin embargo, por muchos ejemplos de derivaciones que hayamos visto para una gramática, el paso de generalización a un lenguaje concreto no siempre es trivial. Del mismo modo, si nos piden una gramática que genere un cierto lenguaje incontextual, el hecho de encontrarla puede no ser trivial.


En ambos casos, lo único que nos puede ayudar es adquirir una cierta práctica con una serie de ejemplos. Poco a poco se van conociendo una serie de “técnicas” y “metodologías”: la utilización de las variables de una cierta manera, el juego de las operaciones sobre lenguajes, etc.

Consultad los ejercicios de autoevaluación al final de este módulo didáctico.



Ahora bien, ¿cómo podemos estar seguros de que lo hemos hecho bien? ¿Cómo nos podemos convencer de que una cierta gramática incontextual genera un determinado lenguaje, o de que un cierto lenguaje incontextual es generado por una determinada gramática? La demostración de estas cuestiones es lo que denominaremos *verificación*.

La **verificación** de una gramática es la demostración de que esta gramática genera un lenguaje determinado.

Como sucede con muchos otros problemas importantes, no hay un método general para resolver este problema; es un problema indecidible. En cada caso es necesario plantear una demostración particular; normalmente se utiliza el razonamiento por inducción. Dependiendo del enunciado, lo más habitual es hacer inducción sobre la longitud de las palabras o inducción sobre la longitud de las derivaciones. 

Daremos un pequeño ejemplo de verificación de una gramática. Quizá la gramática incontextual menos trivial de las que hemos dado, en cuanto al hecho de quedar convencidos de que la gramática genera realmente el lenguaje propuesto, es la que correspondía al lenguaje de las cadenas de paréntesis balanceados o bien formados:

#### Nota

Si queréis, podéis encontrar otros ejemplos de verificación de gramáticas en los libros de la bibliografía.

$$S \rightarrow \lambda \mid (S) \mid SS.$$

Para hacer una demostración formal, debemos empezar caracterizando el lenguaje. Podemos decir que una cadena de paréntesis está balanceada si se dan las dos condiciones siguientes:

- a) El número de paréntesis de abrir es igual al de paréntesis de cerrar.
- b) En todos los prefijos de la palabra, el número de paréntesis de abrir es mayor o igual que el número de paréntesis de cerrar.

La primera condición nos asegura que cada paréntesis de abrir tiene su pareja de cerrar, y la segunda condición impide que en un cierto momento no haya más cerrados que abiertos y que no “cuadren” entre sí.

Más formalmente, lo escribiremos de la manera siguiente:

$$L = \{w \in \{(\,)\}^* \mid |w|_(\, = |w|_{)} \wedge \forall p, r \mid w = pr \Rightarrow |p|_(\, \geq |p|_{)}\}.$$

Debemos demostrar las dos afirmaciones siguientes:

- 1) Toda palabra generada por la gramática pertenece al lenguaje (satisface las dos condiciones anteriores).
- 2) Toda palabra que pertenece al lenguaje se puede generar con la gramática.

Iniciamos, pues, la demostración de los dos puntos anteriores:

- 1) Lo demostraremos por inducción sobre la longitud de la derivación:

- a) En esta ocasión, el caso base es una derivación de cero pasos, y por definición, una derivación de cero pasos a partir de  $S$  nos deja la misma  $S$ :

$$S \xRightarrow{0} \alpha = S.$$

Por tanto, de forma obvia  $S$  verifica las dos condiciones.

- b) Supongamos que debemos descomponer la derivación  $(n + 1)$ -ésima en dos partes: la derivación  $n$ -ésima y el siguiente paso de derivación (un solo paso). Entonces, procederemos así:

$$S \xRightarrow{n} \beta \xRightarrow{1} \alpha.$$

Para la hipótesis de inducción,  $\beta$  satisface las dos condiciones. Debemos distinguir tres casos correspondientes a las tres reglas de producción de la gramática que se pueden aplicar en este último paso para derivar  $\alpha$  de  $\beta$ . Debemos ver que en cada caso se cumplen las dos condiciones.

En la primera y última regla ( $S \rightarrow \lambda$  y  $S \rightarrow SS$ ), puesto que no aparecen paréntesis en la parte derecha, es fácil ver que si las dos condiciones se cumplían

#### Recordad que...

... una demostración por inducción consta de dos pasos:

a) demostrar que la afirmación es cierta para un caso base concreto;

b) suponiendo que la afirmación sea cierta para un caso general  $n$  (hipótesis de inducción), demostrar que también es cierta para el caso  $n + 1$ .

antes de aplicar la regla (hipótesis de inducción), también se cumplirán después.

Sólo falta ver cómo afecta el hecho de aplicar la segunda regla ( $S \rightarrow (S)$ ). Puesto que añadimos un paréntesis de abrir y uno de cerrar, está claro que se sigue cumpliendo la primera condición (habrá uno más de cada; por tanto, el mismo número).

En relación con la segunda condición, es necesario considerarla con más detalle. La situación es la siguiente: tenemos una palabra  $\beta$  que cumple la segunda condición y que contiene como mínimo una variable  $S$ , y queremos saber si, al aplicar la regla  $S \rightarrow (S)$  sobre cualquiera de estas variables  $S$  de  $\beta$  (para producir  $\alpha$ ), se puede continuar afirmando que se cumple la segunda condición. Es decir, sabiendo que todos los prefijos de  $\beta$  tienen más paréntesis "(" que ")", se puede demostrar que después de aplicar la regla indicada obtenemos  $\alpha$ , que también tiene más paréntesis "(" que ")". Entonces, descompondremos las dos palabras:  $\beta = \beta_1 S \beta_2$  y  $\alpha = \beta_1 (S) \beta_2$ .

También en este caso es necesario distinguir los siguientes tres subcasos de prefijos en  $\alpha$ :

- Los prefijos de  $\beta_1$ : coinciden en las dos palabras, y por hipótesis de inducción cumplen la segunda condición.
- Los prefijos de  $\beta_1(S$  que no lo son de  $\beta_1$ : son dos,  $\beta_1 ($  y  $\beta_1(S$ , que tienen un paréntesis más de abrir que  $\beta_1$ , que es prefijo de  $\beta$  y, por tanto, también cumplen la segunda condición.
- Los prefijos de  $\beta_1(S)\beta_2$  que no lo son de  $\beta_1(S$ : en todos éstos añadimos un paréntesis de abrir y uno de cerrar a los prefijos de  $\beta$  y, por tanto, no alteran la relación de la segunda condición.

Con esto queda demostrado que toda palabra generada por la gramática es balanceada y, por tanto, pertenece al lenguaje.

2) Lo demostraremos por inducción sobre la longitud de la palabra (es necesario ver que toda palabra del lenguaje se puede generar con esta gramática):

a) El caso base es la palabra de longitud 0, la palabra vacía. Y la palabra vacía se puede derivar en un solo paso con la regla  $S \rightarrow \lambda$ .

b) En la hipótesis de inducción suponemos que toda palabra de longitud menor o igual que  $n$  se puede generar con la gramática, y tenemos que demostrar que una palabra de longitud  $n + 1$  también se puede generar.

Sea la palabra  $w \neq \lambda$  tal que  $w = pr$ , donde  $p$  es el menor prefijo no vacío de  $w$  que pertenece al lenguaje (existe seguro, pero no será la misma  $w$ ). Se pueden dar dos casos:

- $p$  es un prefijo propio ( $p \neq w$ ) de  $w$ : si  $p$  pertenece al lenguaje, también pertenece el resto de la palabra  $r$  (observad con detalle las dos condiciones que caracterizaban una cadena balanceada). Puesto que  $p$  y  $r$  son no vacíos, presentan una longitud menor que  $w$ , y por hipótesis de inducción se pueden generar con la gramática:

$$S \xRightarrow{*} p \quad \text{y} \quad S \xRightarrow{*} r$$

Esto significa que la palabra  $w$  se puede generar con la siguiente secuencia de derivaciones:


$$S \Rightarrow SS \xRightarrow{*} pr = w.$$

- $p$  no es un prefijo propio de  $w$  ( $p = w$ ) y la palabra  $w$  sólo puede ser de la forma  $w = (v)$  con  $v \in L$ , por hipótesis de inducción  $S \xRightarrow{*} (v)$  (se puede generar porque es de longitud menor que  $w$ ) y, por tanto, la palabra  $w$  se puede generar con la siguiente secuencia de derivaciones:

$$S \Rightarrow (S) \xRightarrow{*} (v)$$

Con esto queda demostrado que toda palabra del lenguaje se puede generar con la gramática propuesta.

A pesar de que es una demostración corta y sencilla, ya nos da una idea clara del tipo de pasos que hay que seguir para verificar una gramática. Además, hemos podido ver un ejemplo de inducción sobre la longitud de las palabras y otro de inducción sobre la longitud de las derivaciones, que son los más habituales en estos casos.

Los problemas de una demostración surgen cuando la gramática juega con muchas variables y muchas reglas de producción; entonces, hay una explosión de casos. Por este motivo, en el próximo apartado trataremos de la simplificación o depuración de gramáticas, paso previo a su verificación. De todos modos, en esta asignatura nos interesa más la verificación de gramáticas desde el punto de vista práctico, como metodología para asegurarnos de que la gramática que proponemos genera el lenguaje dado. No es preciso escribirlo todo de manera formal, pero sí lo es seguir todos los pasos que seguiría un razonamiento por inducción. 

## 4. Simplificación de una gramática

Ya sabemos que todo lenguaje incontextual puede tener diferentes gramáticas que lo generen. De momento, hemos diferenciado las gramáticas ambiguas de las inambiguas, y nos hemos decantado por las primeras siempre que sea posible.

Observad la diferenciación de las gramáticas ambiguas y las inambiguas en el apartado 2 de este módulo didáctico.

Ahora bien, una vez establecida la gramática, puede ser más o menos complejo saber lo que hace (o lo que sería igual, verificarla), dependiendo de si contiene o no elementos “inútiles”, es decir, símbolos o reglas que no le aportan nada.

El proceso de eliminación de elementos (reglas o símbolos) inútiles de una gramática recibe el nombre de **simplificación** o **depuración de la gramática**.

Hay que tener en cuenta que la simplificación no significa necesariamente “reducir” o hacer menor la gramática; algunas veces incluso “crece” exteriormente. Lo que se pretende con la simplificación es eliminar aquellos elementos que nos molestan para posteriores aplicaciones en las que tengamos que utilizar la gramática\*.

\* Por ejemplo, en algoritmos que tienen que decidir sobre la pertenencia de una palabra al lenguaje generado por la gramática.

### 4.1. Gramática limpia

En primer lugar, veremos cómo se pueden eliminar reglas inútiles, en concreto las reglas de producción vacías ( $A \rightarrow \lambda$ , donde  $A$  es una variable) y las reglas de producción unitarias ( $A \rightarrow B$ , donde  $A$  y  $B$  son variables). La gramática resultante se denomina **gramática limpia**, y el proceso de eliminación, **limpieza**.

Una gramática incontextual está limpia si no tiene reglas del tipo  $A \rightarrow \lambda$  (vacías) ni del tipo  $A \rightarrow B$  (unitarias).

Está claro que si la gramática generaba la palabra vacía, al limpiarla dejará de generarla. Sin embargo, exceptuando la palabra vacía, la gramática continúa generando el mismo lenguaje.

Para todo lenguaje incontextual hay una gramática limpia que genera el mismo lenguaje sin la palabra vacía.

El proceso de limpieza se debe hacer en dos partes, una primera de eliminación de las reglas vacías y una segunda de eliminación de las reglas unitarias, exactamente en este orden. Veámoslo:

1) **Eliminación de reglas de producción vacías:** se trata de añadir directamente en las reglas lo que se deja de generar al eliminar las reglas vacías. El método para hacerlo consta de tres pasos:

a) Se determinan las variables anulables, es decir, aquellas que pueden derivar la palabra vacía ( $A \xRightarrow{*} \lambda$ ).

b) Se amplían las reglas en las que aparecen variables anulables en la parte derecha añadiendo una regla idéntica, pero sin la variable anulable. En el caso de que haya más de una variable anulable en la parte derecha, es necesario añadir tantas reglas como combinaciones posibles haya, sustituyendo cada variable anulable por la palabra vacía. Ni que decir tiene que, si así generamos reglas que ya están, no es preciso añadirlas.

#### Ampliación de las reglas

Si hay una variable anulable, añadimos una regla; si hay dos, añadimos tres (si son la misma, dos), etc.

c) Finalmente, se eliminan las reglas vacías.

De esta manera conseguimos una gramática equivalente a la gramática inicial (exceptuando la palabra vacía), pero sin reglas vacías (es decir, limpia).

#### Ejemplo de eliminación de reglas de producción vacías

Supongamos la siguiente gramática incontextual:

$$\begin{aligned} S &\rightarrow aBCb \mid BB \mid aC, \\ A &\rightarrow aAbB \mid Ca, \\ B &\rightarrow \lambda \mid aCS \mid B, \\ C &\rightarrow bSBa \mid AA. \end{aligned}$$

Para conseguir una gramática sin reglas vacías, daremos los siguientes pasos:

a) Las variables anulables son  $B$  (directamente) y  $S$  (indirectamente, por medio de la regla  $BB$ ).

b) Añadiendo las reglas de la ampliación al final, obtenemos la siguiente gramática:

$$\begin{aligned} S &\rightarrow aBCb \mid BB \mid aC \mid aCb \mid B \mid \lambda, \\ A &\rightarrow aAbB \mid Ca \mid aAb, \\ B &\rightarrow \lambda \mid aCS \mid B \mid aC, \\ C &\rightarrow bSBa \mid AA \mid bSa \mid bBa \mid ba. \end{aligned}$$

c) Si eliminamos las reglas vacías, conseguimos la siguiente gramática, que es equivalente a la inicial, excepto en la generación de la palabra vacía:

$$\begin{aligned} S &\rightarrow aBCb \mid BB \mid aC \mid aCb \mid B, \\ A &\rightarrow aAbB \mid Ca \mid aAb, \\ B &\rightarrow aCS \mid B \mid aC, \\ C &\rightarrow bSBa \mid AA \mid bSa \mid bBa \mid ba, \end{aligned}$$

2) **Eliminación de reglas unitarias:** recordemos que las reglas unitarias son del tipo  $A \rightarrow B$ , donde  $A$  y  $B$  son dos variables cualesquiera. Para eliminarlas distinguiremos dos tipos: aquéllas en las que  $A$  y  $B$  son “equivalentes” (más fá-



ciles de eliminar) y aquéllas en las que no lo son (se deben eliminar en un cierto orden). Veamos cómo hay que identificar cada tipo y cómo tenemos que eliminarlas:

a) Diremos que dos variables son equivalentes si de cada una podemos derivar (en cualquier número de pasos) a la otra. Más formalmente,  $A$  y  $B$  son equivalentes si se cumple:

$$A \xRightarrow{*} B \quad \text{y} \quad B \xRightarrow{*} A.$$

Es bastante evidente que las reglas unitarias  $A \rightarrow B$ , en las que  $A$  y  $B$  son variables equivalentes, se pueden eliminar sin que esto afecte de ninguna manera al lenguaje generado por la gramática (conseguimos una gramática equivalente)\*. Por la misma definición de equivalencia se puede derivar cada una de las reglas unitarias a partir de la otra, y no perdemos potencia de generación por el hecho de no poder derivar en un solo paso la una de la otra.

\* Si volvéis a la definición de lenguaje generado por la gramática, os convenceréis de ello.

De hecho, una vez establecidas todas las equivalencias entre todas las variables, nos podemos quedar con un representante de cada clase de equivalencia y reescribir la gramática reemplazando cada variable por su representante, y eliminando las reglas repetidas además de las del tipo  $A \rightarrow A$ .

b) Una vez juntadas las variables equivalentes, debemos ordenar las variables restantes de manera que  $A < B$  si  $A$  se puede derivar de  $B$  (en cualquier número de pasos mayor que cero). Nos referiremos a las variables con subíndices de la siguiente manera:  $A_1 < A_2 < \dots < A_n$ .

- Cuando disponemos de este orden, por definición no puede haber ninguna regla unitaria con la parte izquierda  $A_1$ .
- Si las reglas con parte izquierda  $A_2$  tienen alguna regla unitaria, por definición debe ser  $A_2 \rightarrow A_1$ , y ésta se puede eliminar si a la vez añadimos todas las reglas en las que  $A_1$  es parte izquierda, pero con  $A_2$  como nueva parte izquierda.
- Si las reglas con parte izquierda  $A_3$  tienen alguna regla unitaria, por definición debe ser  $A_3 \rightarrow A_1$  o bien  $A_3 \rightarrow A_2$ , y ésta se puede eliminar si a la vez añadimos todas las reglas en las que  $A_1$  y  $A_2$  son parte izquierda, pero con  $A_3$  como nueva parte izquierda.

#### Cuando decimos...

... "por definición", nos referimos a la definición que hemos dado del orden.

#### Nota

Ya os habíamos adelantado que una gramática limpia no necesariamente "ocupa" menos espacio, ya que aunque se elimina un tipo de reglas, para hacerlo hay que repetir alguna información en varios lugares.

Procederemos así sucesivamente hasta eliminar todas las producciones unitarias para conseguir nuestro objetivo.

### Ejemplo de eliminación de reglas unitarias

Supongamos la siguiente gramática:

$$\begin{aligned} S &\rightarrow aBCb \mid B \mid aC, \\ A &\rightarrow aAbB \mid C, \\ B &\rightarrow S \mid aCS \mid B \mid C, \\ C &\rightarrow bSBa \mid AA. \end{aligned}$$

Nuestro proceso de eliminación de las reglas unitarias sería el siguiente:

a) Con una simple ojeada identificamos dos variables equivalentes,  $S$  y  $B$ . También aparecen otras dos reglas unitarias,  $A \rightarrow C$  y  $B \rightarrow C$ , pero puesto que de la variable  $C$  no se puede llegar a derivar ni  $A$  ni  $B$ , estas reglas quedan pendientes de eliminación hasta el segundo paso. Eliminamos la variable  $B$ , cambiamos todas las  $B$  por  $S$  y juntamos todas las reglas con parte izquierda  $S$  o  $B$ . Entonces obtenemos la siguiente gramática:

$$\begin{aligned} S &\rightarrow aSCb \mid aC \mid aCS \mid C, \\ A &\rightarrow aAbS \mid C, \\ C &\rightarrow bSSa \mid AA. \end{aligned}$$

b) Ahora debemos ordenar las variables restantes ( $S$ ,  $A$  y  $C$ ), según el criterio indicado, para poder iniciar el proceso de eliminación de las dos reglas unitarias que quedan. En nuestro ejemplo hay dos ordenaciones posibles compatibles con la definición, que son las siguientes:

$$C < A < S \text{ o bien } C < S < A.$$

Tanto a partir de  $A$  como de  $S$  podemos llegar a derivar  $C$  y, por tanto,  $C$  debe ser la palabra más pequeña. Dado que  $A$  no puede llegar a generar  $S$  ni  $S$  puede llegar a generar  $A$ , es indiferente el orden en que las pongamos.

Como decíamos, observamos que las reglas con la variable menor,  $C$ , en la parte izquierda ya son correctas, no hay ninguna variable unitaria.


Si elegimos el primer orden dado, la segunda variable es  $A$ . Hay una regla con  $A$  como parte izquierda, que necesariamente debía ser  $A \rightarrow C$ . Para poderla eliminar debemos incorporar las reglas con  $C$  en la parte izquierda, pero cambiando  $C$  por  $A$ . Así, obtenemos:

$$\begin{aligned} S &\rightarrow aSCb \mid aC \mid aCS \mid C, \\ A &\rightarrow aAbS \mid bSSa \mid AA, \\ C &\rightarrow bSSa \mid AA. \end{aligned}$$

Ahora tenemos que hacer lo mismo para eliminar la última regla unitaria,  $S \rightarrow C$ , incorporando las reglas con  $C$  en la parte izquierda, pero con  $S$  en lugar de  $C$ :

$$\begin{aligned} S &\rightarrow aSCb \mid aC \mid aCS \mid bSSa \mid AA, \\ A &\rightarrow aAbS \mid bSSa \mid AA, \\ C &\rightarrow bSSa \mid AA. \end{aligned}$$

Hemos obtenido una gramática equivalente a la inicial, pero sin reglas de producción unitarias. Teniendo en cuenta que esto se haría después de eliminar las reglas vacías (nuestro ejemplo ya no las tenía inicialmente), obtenemos una gramática limpia.

Las gramáticas incontextuales limpias tienen mucha importancia al estudiar el problema de la pertenencia de una palabra a un lenguaje generado por una gramática incontextual. Así, los algoritmos para decidir si una palabra  $m \in L(G)$  trabajan sobre gramáticas limpias son más eficientes, porque hay un límite superior en el número de derivaciones de un solo paso necesarias para generar cualquier palabra. 

#### Observad que...

... también se elimina la regla unitaria  $B \rightarrow B$ , que ya se podía haber quitado sin hacer nada más.

#### Observad que...

... por ejemplo, las reglas vacías y unitarias pueden introducir ciclos en la derivación.

## 4.2. Gramática pelada

En la “limpieza” de gramáticas incontextuales ponemos restricciones (eliminación de reglas vacías y reglas unitarias) que no afectan al poder de expresión

de las gramáticas. De hecho, hemos dado pautas para realizar estas eliminaciones sin que afecten al lenguaje que genera la gramática. Básicamente, la limpieza consistía en realizar un cambio de “formato”.

Nos interesaba que no hubiera ciertos tipos de reglas, pero no las podíamos quitar así como así porque podían cumplir una cierta función. Lo debíamos contrarrestar cubriendo estas funciones con otro tipo de reglas. Simplemente, hemos demostrado que esto era posible.

Ahora bien, también puede darse el caso de que haya elementos “inútiles” de verdad: variables o reglas que no aporten nada a la gramática, que quitándolas sin más no cambie el conjunto de palabras o lenguaje generado por la gramática. Este proceso se denomina **pelar una gramática**, y la gramática resultante, **gramática pelada**.


Una gramática incontextual está pelada si de todas sus variables se puede derivar alguna palabra (coaccesible) y, además, si todas sus variables aparecen en algún paso de alguna derivación a partir de  $S$  (accesible).

De manera más formal, podemos decir que si una gramática está pelada, se cumple que:

- 1)  $\forall A \in V \exists m \in \Sigma^*$  tal que  $A \xRightarrow{*} m$ .
- 2)  $\forall A \in V \exists \alpha, \beta \in (\Sigma \cup V)^*$  tal que  $S \xRightarrow{*} \alpha A \beta$ .


#### Nota

Observad los parecidos entre las gramáticas peladas y lo que en el módulo “Autómatas finitos y lenguajes regulares” hemos denominado *autómata finito pelado*, aquél que no tiene estados inútiles.

Fijaos en que sólo decimos que una gramática está pelada si no contiene variables que no participen en la generación de palabras del lenguaje que genera, ya sea porque nos impiden derivar una palabra sólo formada por símbolos terminales (primera condición) o bien porque no pueden llegar a salir nunca en una derivación a partir de la variable inicial (segunda condición). 


El hecho de eliminar estas variables y las reglas en que aparecen como parte izquierda\* no afecta al lenguaje generado por la gramática. Las primeras, porque si aparecen en el curso de una derivación aseguran que no se pueda llegar a generar una palabra (por tanto, mejor que no salgan) y las segundas, porque no pueden salir (por tanto, sus reglas de producción no son necesarias, ya que nunca se aplicarán).

\* Además de desaparecer de todas las partes derechas de las reglas restantes, claro.

Lo realmente importante no es saber cuándo una gramática está pelada, sino que toda gramática se puede pelar sin que esto afecte al lenguaje que genera. 

Por cada gramática incontextual que genera un lenguaje no vacío hay una gramática pelada que genera el mismo lenguaje.

Lo único que decimos es que las variables que no pueden acabar transformándose en una palabra de símbolos terminales (primera condición) y las que no aparecen nunca en una derivación (segunda condición) son inútiles y se pueden eliminar.

Describiremos un método sistemático para identificar estas variables inútiles y así poder pelar la gramática. Consta de dos pasos: 

1) Identificar las variables a partir de las cuales se puede derivar una palabra de símbolos terminales.

Una posibilidad es construir un conjunto de variables de forma incremental: inicialmente, es necesario identificar las variables que derivan, en un solo paso, una palabra con sólo símbolos terminales (reglas que tienen la variable en la parte izquierda y una palabra sin variables en la parte derecha). A continuación, debemos añadirle las variables que en un solo paso derivan una palabra formada por símbolos terminales y las variables anteriores, y así sucesivamente hasta que ya no podamos incorporar ninguna nueva variable (porque están todas o bien porque las que quedan no se pueden incorporar).

2) A partir de las variables resultantes del paso anterior, debemos seleccionar aquellas que aparecen en alguna derivación a partir de  $S$ .

De manera parecida al paso anterior (construcción incremental de un conjunto de variables), partimos inicialmente de la variable inicial  $S$ , a continuación añadimos las variables que aparecen en la parte derecha de una regla con  $S$  en la parte izquierda, después añadimos las variables que aparecen en la parte derecha de una regla con una de las variables anteriores en la parte izquierda, y así sucesivamente hasta que no podamos añadir ninguna otra nueva.

Lo importante es la definición de gramática pelada, ya que hay otras maneras de identificar las variables inútiles. Nosotros sólo hemos proporcionado una manera sistemática de las muchas posibles. Después de los dos pasos de identificación de variables útiles, debemos eliminar todas las variables inútiles de la gramática con todo lo que esto comporta: eliminar todas las reglas de producción en que aparece alguna de estas variables tanto en su parte derecha como en su parte izquierda.

Veámoslo con un ejemplo sobre la siguiente gramática incontextual:

$$\begin{aligned} S &\rightarrow aAb \mid bDA \mid AEb, \\ A &\rightarrow aaA \mid Caba \mid DCba, \\ B &\rightarrow bAB \mid aaD, \\ C &\rightarrow \lambda \mid Aa, \\ D &\rightarrow baba \mid AD, \\ E &\rightarrow EbbCB, \\ F &\rightarrow \lambda \mid bbB \mid aaA. \end{aligned}$$

#### Observad que...

... se puede llevar a cabo este paso sólo considerando las diferentes reglas de producción con la ayuda de un conjunto de variables que va creciendo.

#### Decimos que...

... el proceso finaliza cuando entramos en un **estado estacionario**.

#### Nota

Observad que si la variable inicial  $S$  es inútil, esto significa que la gramática genera el lenguaje vacío (aquel que no tiene ninguna palabra).

a) Indicaremos cómo evoluciona el conjunto de variables útiles paso a paso según el método indicado:

- $\{C, D, F\}$ ,
- $\{A, B, C, D, F\}$ ,
- $\{S, A, B, C, D, F\}$ .

Esto significa que la variable  $E$  es inútil, porque no se puede llegar a ninguna palabra de símbolos finales a partir de ésta. Después de las eliminaciones nos queda la siguiente gramática:

$$\begin{aligned} S &\rightarrow aAb \mid bDA, \\ A &\rightarrow aaA \mid Caba \mid DCba, \\ B &\rightarrow bAB \mid aaD, \\ C &\rightarrow \lambda \mid Aa, \\ D &\rightarrow baba \mid AD, \\ F &\rightarrow \lambda \mid bbB \mid aaA. \end{aligned}$$

b) Damos también la evolución del conjunto de variables útiles, que son las siguientes:

- $\{S\}$ ,
- $\{S, A, D\}$ ,
- $\{S, A, C, D\}$ .

Llegamos a la conclusión de que las variables  $B$  y  $F$  son inútiles. Si las eliminamos junto con las reglas en que aparecen, obtenemos la siguiente gramática pelada:

$$\begin{aligned} S &\rightarrow aAb \mid bDA, \\ A &\rightarrow aaA \mid Caba \mid DCba, \\ C &\rightarrow \lambda \mid Aa, \\ D &\rightarrow baba \mid AD. \end{aligned}$$

## 5. Formas normales

Las simplificaciones de gramáticas incontextuales del apartado anterior pretendían principalmente eliminar todos los elementos que podían molestar por varias razones:

- Las **reglas de producción vacías y unitarias** (limpieza), porque podían provocar ciclos en las derivaciones que hicieran difícil la identificación de palabras que pertenecen al lenguaje generado por la gramática.
- Los **símbolos inútiles** (pelar) porque podían llevar a confusión en la comprensión de lo que hacía la gramática\*.

En cierta manera, estas simplificaciones\* se pueden considerar también normalizaciones (algunos libros las presentan así).

\* Por ejemplo, de cara a una verificación más o menos formal.

\* Algunos libros prefieren hablar de depuración, de eliminación de impurezas, etc.

Diremos que una gramática está en **forma normal** si sus variables y reglas están sujetas a restricciones especiales, sin que ello afecte al lenguaje que ha generado.

Las dos formas normales más utilizadas en gramáticas incontextuales son la **forma normal de Chomsky** y la **forma normal de Greibach**.

Ya veremos que, de alguna manera, utilizan las simplificaciones del apartado anterior, por lo que son más restrictivas.

### 5.1. Forma normal de Chomsky

Una gramática incontextual está en **forma normal de Chomsky** si todas sus reglas de producción están en una de las dos formas siguientes:

- 1)  $A \rightarrow BC$ ,
- 2)  $A \rightarrow a$ ,

donde,  $A, B, C \in V$  (son variables) y  $a \in \Sigma$  (es un símbolo terminal).

#### Observad que...

... si una gramática está en forma normal de Chomsky, por definición está limpia (no puede tener ni reglas vacías ni unitarias).

Por ejemplo, la siguiente gramática está en forma normal de Chomsky:

$$\begin{aligned} S &\rightarrow AB \mid AC \mid SS, \\ C &\rightarrow SB, \\ A &\rightarrow (, \\ B &\rightarrow ). \end{aligned}$$

Esta gramática genera el lenguaje de las cadenas no nulas de paréntesis balanceados.

Lo primero que notamos es que una gramática en forma normal de Chomsky no puede generar la palabra vacía (lo mismo que nos pasaba con una gramática limpia). Sin embargo, ésta es la única excepción\*.

\* Para evitar esta excepción a veces se permite una sola regla vacía,  $S \rightarrow \lambda$ .


Por cada gramática incontextual  $G$  hay una gramática en forma normal de Chomsky  $G'$  que genera el mismo lenguaje, excepto la palabra vacía. Es decir:

$$L(G') = L(G) - \{\lambda\}.$$

Observamos que el árbol de derivación de cualquier palabra generada por una gramática incontextual en forma normal de Chomsky es un árbol binario: los nodos intermedios tendrán exactamente dos hijos incluyendo la raíz (reglas de tipo 1), y las hojas, un solo hijo (reglas de tipo 2).

Podéis ver los árboles en la asignatura *Estructura de la información*.

Esta observación nos permite introducir un método para pasar cualquier gramática incontextual a una forma normal de Chomsky: hacer lo mismo que cuando queremos transformar un árbol general en un árbol binario.

A continuación, detallamos los pasos de la transformación a forma normal de Chomsky: 

- 1) Limpiamos la gramática (es decir, eliminamos las reglas de producción vacías y unitarias).
- 2) Para cada símbolo terminal  $a \in \Sigma$ , introducimos una nueva variable  $A_a$  y la regla  $A_a \rightarrow a$ , y reemplazamos todas las apariciones de  $a$  en la parte derecha de todas las reglas (excepto si la regla es de tipo 2,  $A \rightarrow a$ ) por  $A_a$ .

Después de estas sustituciones, todas las reglas serán de la forma  $A \rightarrow a$  o bien  $A \rightarrow B_1 B_2 \dots B_k$  con  $k \geq 2$  y  $B_i$  variables. Las primeras ya son de tipo 2 y las segundas con  $k = 2$  ya son de tipo 1.

También queda claro que el conjunto de palabras generadas no ha cambiado, simplemente ahora hemos tardado un paso más de derivación en generar cada símbolo terminal.

- 3) Por cada regla  $A \rightarrow B_1 B_2 \dots B_k$  con  $k \geq 3$ , introducimos una nueva variable  $C$  y reemplazamos esta regla por las dos reglas siguientes\*:

$$\begin{aligned} A &\rightarrow B_1 C, \\ C &\rightarrow B_2 \dots B_k; \end{aligned}$$

\* Repetimos este proceso hasta que todas las partes derechas de las reglas tengan como máximo dos variables (longitud 2)

Con el método de transformación a forma normal de Chomsky tampoco cambia el lenguaje generado, sólo hemos agrupado por parejas las partes derechas de las reglas mediante la introducción de nuevas variables. Veámoslo con un par de ejemplos (estos ejemplos son sobre gramáticas ya limpias; así pues, no es necesario aplicar el primer paso):

1) Hallamos una gramática en forma normal de Chomsky para el lenguaje  $L = \{a^n b^n \mid n \geq 1\}$ .

Ya habíamos dado una gramática para este lenguaje, si bien ahora no contiene la palabra vacía, que era la siguiente:

$$S \rightarrow aSb \mid ab.$$

Comenzamos a buscar la forma normal de Chomsky (recordad que no es necesario aplicar el primer paso):

a) Añadimos las variables  $A$  y  $B$  para sustituir  $a$  y  $b$ , así como las reglas correspondientes, y obtenemos la siguiente gramática:

$$\begin{aligned} S &\rightarrow ASB \mid AB, \\ A &\rightarrow a, \\ B &\rightarrow b. \end{aligned}$$

b) Añadimos la variable  $C$  para dividir en dos la primera regla y ya la tenemos en forma normal de Chomsky:

$$\begin{aligned} S &\rightarrow AC \mid AB, \\ C &\rightarrow SB, \\ A &\rightarrow a, \\ B &\rightarrow b. \end{aligned}$$

2) Hallamos la forma normal de Chomsky para la siguiente gramática:

$$\begin{aligned} S &\rightarrow aSA \mid SB, \\ A &\rightarrow bAA \mid a, \\ B &\rightarrow aB \mid b. \end{aligned}$$

Igual que antes, obviamos el primer paso, ya que partimos de una gramática limpia:

a) Añadimos las variables  $C$  y  $D$  para sustituir  $a$  y  $b$  (sólo si no están solas), junto con las reglas  $C \rightarrow a$  y  $D \rightarrow b$ :

$$\begin{aligned} S &\rightarrow CSA \mid SB, \\ A &\rightarrow DAA \mid a, \\ B &\rightarrow CB \mid b, \\ C &\rightarrow a, \\ D &\rightarrow b. \end{aligned}$$



b) Debemos introducir dos nuevas variables ( $E$ ,  $F$ ) para dividir en dos las dos partes derechas con tres variables, y obtenemos la siguiente gramática:

$$\begin{aligned} S &\rightarrow CE \mid SB, \\ E &\rightarrow SA, \\ A &\rightarrow DF \mid a, \\ F &\rightarrow AA, \\ B &\rightarrow CB \mid b, \\ C &\rightarrow a, \\ D &\rightarrow b. \end{aligned}$$

## 5.2. Forma normal de Greibach

Una gramática incontextual está en **forma normal de Greibach** si todas sus reglas de producción son de la forma:

$$A \rightarrow a\alpha,$$

donde  $A \in V$  es una variable,  $a \in \Sigma$  es un símbolo terminal y  $\alpha \in V^*$  es una concatenación de variables.


### Forma normal de Greibach

Dicho de otra manera, cada regla tiene la forma  $A \rightarrow aB_1B_2 \dots B_k$  para  $k \geq 0$ , donde  $A, B_1, B_2, \dots, B_k$  son variables, y  $a$ , un símbolo terminal.

Por ejemplo, la siguiente gramática está en forma normal de Greibach y genera el lenguaje de las cadenas no nulas de paréntesis balanceados:

$$\begin{aligned} S &\rightarrow (B \mid (SB \mid (BS \mid (SBS, \\ B &\rightarrow ), \end{aligned}$$

Comparad esta gramática con la que hemos dado, en el subapartado 5.1. de este módulo didáctico, en forma normal de Chomsky para el mismo lenguaje.

Al igual que sucedía en las gramáticas en forma normal de Chomsky, una gramática en forma normal de Greibach tampoco puede generar la palabra vacía, según su definición (en toda parte derecha de la regla hay un símbolo terminal). 

A pesar de que intuitivamente no está tan claro como en la forma normal de Chomsky, a parte de la palabra vacía, siempre se puede pasar una gramática a forma normal de Greibach.

Para toda gramática incontextual  $G$  existe una gramática en forma normal de Greibach  $G'$  que genera el mismo lenguaje excepto la palabra vacía:


$$L(G') = L(G) - \{\lambda\}.$$

El interés de la forma normal de Greibach es que el número de símbolos terminales al principio de la palabra en curso se incrementa en cada paso de una

derivación “más a la izquierda” (es decir, en la que siempre derivamos la variable situada más a la izquierda).

Pasamos a ver un método para transformar cualquier gramática incontextual a forma normal de Greibach: 

1) En primer lugar, pasamos la gramática a forma normal de Chomsky (sólo quedan por tratar las reglas de tipo 1). A continuación, reescribimos la gramática obtenida cambiando el nombre de las variables, numerándolas con la ayuda de los subíndices:  $A_1, A_2, \dots, A_k$ .

 Recordad el método para generar una forma normal de Chomsky, que hemos visto en el subapartado 5.1. de este módulo didáctico.

2) A continuación, hacemos los cambios necesarios de manera que las reglas  $A_i \rightarrow A_j \alpha$  cumplan  $j > i$ , comenzando con las reglas con parte izquierda  $A_1$  y en orden, hasta acabar con las reglas con parte izquierda  $A_k$ . Debemos distinguir dos casos:

- Caso  $i > j$ : para poder eliminar la regla  $A_i \rightarrow A_j \alpha$ , debemos añadir nuevas reglas  $A_i \rightarrow \beta \alpha$ , en las que  $\beta$  son las partes derechas de todas las reglas con  $A_j$  en la parte izquierda. Quizá tendremos que repetir este proceso para alguna de estas nuevas reglas. Acabaremos cuando en todas las reglas  $A_i \rightarrow A_j \alpha$  se cumpla  $i \leq j$ .
- Caso  $i = j$ : en este caso no podemos aplicar el proceso anterior, ya que entraríamos en un bucle infinito. Debemos introducir una nueva variable  $B_i$  para cada caso. Y para poder eliminar la regla  $A_i \rightarrow A_i \alpha$  debemos añadir las reglas  $B_i \rightarrow \alpha$  y  $B_i \rightarrow \alpha B_i$  con la nueva variable como parte izquierda, y las reglas  $A_i \rightarrow \beta B_i$ , donde  $\beta$  es la parte derecha de todas las otras reglas en que  $A_i$  es parte izquierda.

3) Ya estamos en condiciones de dejar todas las reglas con  $A_i$  en la parte izquierda en forma normal de Greibach. En sentido inverso al del paso anterior (de  $A_k$  hasta  $A_1$ ), todas las reglas  $A_i \rightarrow A_j \alpha$  que ahora ya cumplen  $j > i$  deben eliminarse y añadir en su lugar las nuevas reglas  $A_i \rightarrow \beta \alpha$ , donde  $\beta$  es la parte derecha de las reglas en que  $A_j$  es la parte izquierda.

4) El último paso es muy simple. Sólo quedan las variables  $B_i$  (si tenemos que introducir alguna) que pueden no estar en forma normal de Greibach. Las que no lo están, deben tener la forma  $B_i \rightarrow A_j \alpha$ . Y para eliminarlas sólo es necesario sustituirlas por nuevas reglas  $B_i \rightarrow \beta \alpha$ , donde  $\beta$  es la parte derecha de todas las reglas en que  $A_j$  es parte izquierda.

#### Observad que...

... si todas las reglas  $A_i \rightarrow A_j \alpha$  cumplen  $j > i$ , esto significa que la mayor,  $A_k$ , tiene un símbolo terminal al principio de la parte derecha. Así, mediante sucesivas sustituciones, conseguimos que todas las partes derechas comiencen con un símbolo terminal (forma normal de Greibach).

Ahora veremos un par de ejemplos para comprobar que el método es mucho más sencillo de lo que parece después de hacer su descripción semiformal. El primer ejemplo será más sencillo que el segundo, ya que no tiene reglas  $A_i \rightarrow A_i \alpha$  que alarguen el proceso:

1) Hallamos la forma normal de Greibach para la siguiente gramática:

$$\begin{aligned} S &\rightarrow BA, \\ A &\rightarrow a, \\ A &\rightarrow abABa, \\ B &\rightarrow b. \end{aligned}$$

Aplicamos los pasos que forman este método:

a) Si pasamos esta gramática a forma normal de Chomsky y renombramos las variables, obtenemos la siguiente gramática:

$$\begin{aligned} A_1 &\rightarrow A_3A_2, \\ A_2 &\rightarrow a \mid A_4A_5, \\ A_3 &\rightarrow b, \\ A_4 &\rightarrow a, \\ A_5 &\rightarrow A_6A_7, \\ A_6 &\rightarrow b, \\ A_7 &\rightarrow A_2A_8, \\ A_8 &\rightarrow A_3A_9, \\ A_9 &\rightarrow a. \end{aligned}$$

b) Sólo hay dos reglas que no cumplen  $j > i$ . Las tratamos por separado (y a continuación tratamos las nuevas generadas por cada una de éstas):

- $A_7 \rightarrow A_2A_8$  se sustituye por  $A_7 \rightarrow aA_8$  y  $A_7 \rightarrow A_4A_5A_8$ ,
- $A_8 \rightarrow A_3A_9$  se sustituye por  $A_8 \rightarrow aA_5A_8$ ,
- $A_8 \rightarrow A_3A_9$  se sustituye por  $A_8 \rightarrow bA_9$ ,

Ya tenemos una gramática en la que las variables de la parte izquierda de las reglas tienen un subíndice menor que la primera variable de la parte derecha (en caso de ser una variable). Esta gramática es la siguiente:

$$\begin{aligned} A_1 &\rightarrow A_3A_2, \\ A_2 &\rightarrow a \mid A_4A_5, \\ A_3 &\rightarrow b, \\ A_4 &\rightarrow a, \\ A_5 &\rightarrow A_6A_7, \\ A_6 &\rightarrow b, \\ A_7 &\rightarrow aA_8 \mid aA_5A_8, \\ A_8 &\rightarrow bA_9, \\ A_9 &\rightarrow a. \end{aligned}$$

c) Si sustituimos de abajo arriba en todas las reglas que tienen una parte derecha que comience por un símbolo no terminal, obtenemos la siguiente gramática en forma normal de Greibach equivalente a la inicial:

$$\begin{aligned} A_1 &\rightarrow bA_2 \text{ (hemos sustituido } A_3 \text{ por } b), \\ A_2 &\rightarrow a \mid aA_5 \text{ (hemos sustituido } A_4 \text{ por } a), \\ A_3 &\rightarrow b, \\ A_4 &\rightarrow a, \\ A_5 &\rightarrow bA_7 \text{ (hemos sustituido } A_6 \text{ por } b), \\ A_6 &\rightarrow b, \\ A_7 &\rightarrow aA_8 \mid aA_5A_8, \\ A_8 &\rightarrow bA_9, \\ A_9 &\rightarrow a. \end{aligned}$$

Al no haber introducido variables  $B_i$ , el proceso acaba aquí.

2) Encontramos una gramática en forma normal de Greibach para el lenguaje de las cadenas no nulas de paréntesis balanceados:

$$S \rightarrow () \mid (S) \mid SS.$$

Para conseguir nuestra forma normal de Greibach, daremos los pasos siguientes:

a) Anteriormente ya hemos dado una gramática equivalente en forma normal de Chomsky, que era la siguiente:

$$\begin{aligned} S &\rightarrow AB \mid AC \mid SS, \\ A &\rightarrow (, \\ B &\rightarrow ), \\ C &\rightarrow SB. \end{aligned}$$

Consultad la gramática equivalente en el inicio del subapartado 5.1. de este módulo didáctico.

Si numeramos las variables, tenemos:

$$\begin{aligned} A_1 &\rightarrow A_2A_3 \mid A_2A_4 \mid A_1A_1, \\ A_2 &\rightarrow (, \\ A_3 &\rightarrow ), \\ A_4 &\rightarrow A_1A_3. \end{aligned}$$

b) Hay una regla con  $i = j$ ,  $A_1 \rightarrow A_1A_1$ , y para poder eliminarla debemos introducir una nueva variable  $B_1$  y las cuatro reglas siguientes:

$$\begin{aligned} B_1 &\rightarrow A_1, \\ B_1 &\rightarrow A_1B_1, \\ A_1 &\rightarrow A_2A_3B_1, \\ A_1 &\rightarrow A_2A_4B_1. \end{aligned}$$

También hay una regla con  $i > j$ ,  $A_4 \rightarrow A_1A_3$ , que trataremos como antes, sustituyéndola por las reglas (atención a los cambios anteriores en  $A_1$ ):

$$\begin{aligned} A_4 &\rightarrow A_2A_3A_3, \\ A_4 &\rightarrow A_2A_4A_3, \\ A_4 &\rightarrow A_2A_3B_1A_3, \\ A_4 &\rightarrow A_2A_4B_1A_3, \end{aligned}$$

y puesto que en todas se cumple aún  $i > j$ , debemos volver a hacer una sustitución en  $A_2$ , poniendo la parte derecha de su regla, que es  $($ . Así:

$$\begin{aligned} A_4 &\rightarrow (A_3A_3, \\ A_4 &\rightarrow (A_4A_3, \\ A_4 &\rightarrow (A_3B_1A_3, \\ A_4 &\rightarrow (A_4B_1A_3. \end{aligned}$$

En definitiva, después de los cambios la gramática nos queda así:

$$\begin{aligned} A_1 &\rightarrow A_2A_3 \mid A_2A_4 \mid A_2A_3B_1 \mid A_2A_4B_1, \\ A_2 &\rightarrow (, \\ A_3 &\rightarrow ), \\ A_4 &\rightarrow (A_3A_3 \mid (A_4A_3 \mid (A_3B_1A_3 \mid (A_4B_1A_3, \\ B_1 &\rightarrow A_1 \mid A_1B_1. \end{aligned}$$

c) Todas las reglas están en forma normal de Greibach, excepto las que tienen  $A_1$  como parte izquierda (y las que tienen  $B_1$ , que trataremos en el paso siguiente). Y como sus partes derechas comienzan con  $A_2$ , sólo nos falta sustituir ésta por su parte derecha, es decir, por  $($ . La gramática resultante es la siguiente:

$$\begin{aligned} A_1 &\rightarrow (A_3 \mid (A_4 \mid (A_3B_1 \mid (A_4B_1, \\ A_2 &\rightarrow (, \\ A_3 &\rightarrow ), \\ A_4 &\rightarrow (A_3A_3 \mid (A_4A_3 \mid (A_3B_1A_3 \mid (A_4B_1A_3, \\ B_1 &\rightarrow A_1 \mid A_1B_1. \end{aligned}$$


d) Finalmente, para poner en forma normal las reglas de  $B_1$ , debemos sustituir la primera variable de sus partes derechas ( $A_1$  en los dos casos) por todas las partes derechas de las reglas en que  $A_1$  es parte izquierda (las dos que aparecen repetidas ya no las ponemos):

$$\begin{aligned} A_1 &\rightarrow (A_3 \mid (A_4 \mid (A_3B_1 \mid (A_4B_1, \\ A_2 &\rightarrow (, \\ A_3 &\rightarrow ), \\ A_4 &\rightarrow (A_3A_3 \mid (A_4A_3 \mid (A_3B_1A_3 \mid (A_4B_1A_3, \\ B_1 &\rightarrow (A_3 \mid (A_4 \mid (A_3B_1 \mid (A_4B_1 \mid (A_3B_1B_1 \mid (A_4B_1B_1. \end{aligned}$$

La gramática obtenida está claramente en forma normal de Greibach (todas sus reglas comienzan con un símbolo terminal) y es equivalente a la que hemos dado inicialmente.

De todos modos, el método que hemos propuesto no necesariamente nos da la gramática en su forma más simple; al principio ya habíamos dado una gramática más sencilla en forma normal de Greibach para el mismo lenguaje del ejemplo que acabamos de ver, y esta gramática era la que mostramos a continuación:

$$\begin{aligned} S &\rightarrow (B \mid (SB \mid (BS \mid (SBS, \\ B &\rightarrow ). \end{aligned}$$

En este contexto de formas normales, puede ser interesante dar una ojeada al tipo 3 de gramáticas generativas de la clasificación establecida por Chomsky, las llamadas *gramáticas regulares*. Recordad su definición: gramáticas en las que cada regla de producción tiene la forma  $A \rightarrow \alpha B$  o bien  $A \rightarrow \alpha$  con  $\alpha \in \Sigma^*$  y  $A, B \in V$ . 

Las restricciones en las reglas nos hacen pensar en el tipo de restricciones introducidas en las formas normales de Chomsky y de Greibach. En concreto, ambas exigen que la parte derecha de las reglas comience con un símbolo terminal, pero no son idénticas (detrás del símbolo terminal puede ir una sola variable o bien una secuencia de símbolos terminales).

Sin embargo, estas diferencias que acabamos de ver son suficientes para que toda gramática incontextual se pueda expresar con las restricciones de las gramáticas regulares, tal como sucedía en el caso de las formas normales (exceptuando la palabra vacía).

Los lenguajes que se pueden generar con gramáticas con estas restricciones reciben el nombre de **lenguajes regulares**, y a estas gramáticas se las denomina **gramáticas lineales por la izquierda** (además de *gramáticas regulares*).

#### Gramáticas lineales por la derecha

También existen las gramáticas lineales por la derecha, en que la forma de las reglas es  $A \rightarrow B\alpha$  o bien  $A \rightarrow \alpha$  con  $\alpha \in \Sigma^*$  y  $A, B \in V$ . Tienen el mismo poder de generación.

Esto significa que todo lo que hemos dicho y hecho con gramáticas incontextuales también resulta válido para las gramáticas regulares, y que todos los lenguajes estudiados en el módulo “Autómatas finitos y lenguajes regulares” tienen una gramática regular que los genera.

Por ejemplo, el lenguaje regular  $L = a^*b^*$  es generado por la siguiente gramática:

$$\begin{aligned} S &\rightarrow \lambda \mid A \mid B, \\ A &\rightarrow a \mid aA \mid B, \\ B &\rightarrow b \mid bB. \end{aligned}$$

## 6. Autómatas con pila

En otro módulo de esta asignatura ya hemos introducido los autómatas finitos, una máquina abstracta con memoria finita que permite reconocer lenguajes regulares. Todo lenguaje regular tiene un autómata finito que lo reconoce (decide qué palabras pertenecen o no al lenguaje) y todo autómata finito reconoce algún lenguaje regular. Además, como hemos visto en este módulo, todo lenguaje regular es generado por una gramática regular, y toda gramática regular genera un lenguaje regular. Hay una equivalencia entre lenguajes y gramáticas regulares, y autómatas finitos.

Las limitaciones de los autómatas finitos nos permitían comprobar que hay lenguajes que no son regulares, como  $\{a^n b^n\}$ , algunos de los cuales hemos llamado **lenguajes incontextuales** (los que podían ser generados por una gramática de tipo 2, es decir, una gramática incontextual, como el del ejemplo). Intuitivamente, nos hemos dado cuenta de que los lenguajes de este tipo requieren un número infinito de estados (en este caso, para memorizar el número de  $a$  que han sido leídas antes de que se encuentre la primera  $b$ ).

Uno de los tipos más importantes de autómatas infinitos es el que se conoce como autómata con pila.

Un **autómata con pila** es esencialmente un autómata infinito (con control sobre una cinta de entrada, como ya sabéis) que dispone de una memoria (potencialmente infinita) con estructura de pila (LIFO) en la que almacena símbolos de trabajo.


### En la estructura de pila...

... sólo se puede acceder al elemento superior, es decir, al último que ha entrado.

De la misma manera que todos los lenguajes (y gramáticas) regulares tienen un autómata equivalente (el autómata finito), las gramáticas incontextuales también tienen su contrapartida en forma de máquina, que es el autómata con pila. Sin embargo, en este caso, la equivalencia no es tan satisfactoria, ya que el autómata con pila es un dispositivo no determinista y su versión determinista sólo acepta un subconjunto de todos los lenguajes incontextuales.

Consultad la estructura de pila o LIFO (el último que entra es el primero que sale) en la asignatura Estructura de la información.

**Recordad que todo autómata finito no determinista tiene un equivalente determinista.**

En otras palabras, en su versión indeterminista los autómatas con pila son equivalentes a los lenguajes (y gramáticas) incontextuales. Sin embargo, los autómatas con pila deterministas caracterizan sólo un subconjunto estricto de los lenguajes incontextuales. 

A efectos prácticos (como puede ser la compilación), esto no es demasiado importante, porque este subconjunto incluye la sintaxis de la mayoría de los lenguajes de programación.

## 6.1. Definición de autómata con pila

Intuitivamente, un autómata con pila es un autómata finito con  $\lambda$ -transiciones que dispone, además, de una memoria con estructura de pila. Por tanto, su definición debe ser una ampliación de la definición de autómata finito.

Para definirlo, necesitamos un alfabeto de símbolos que pueden ir a la pila (normalmente coincide o es un subconjunto del alfabeto de símbolos de entrada), y un símbolo especial para indicar el fondo de la pila (pila vacía). Además, debemos modificar el conjunto de transiciones para indicar cómo participa la pila.

Una **pila** es una secuencia de elementos accesibles por un solo punto, que denominamos **cima de la pila**\*. Las principales operaciones que podemos hacer con esta estructura es añadir un nuevo elemento (**apilar**), que se coloca sobre la cima de la pila y pasa a ser la nueva cima, y quitar uno (**desapilar**), que elimina la cima de la pila y el elemento que tiene debajo pasa a ser la nueva cima.

\* Es útil representar los elementos de la pila colocados unos sobre otros; el de más arriba es la cima.

Un **autómata con pila** es una estructura formada por los siguientes componentes:

- 1) Un conjunto finito de estados  $Q$ .
- 2) Un alfabeto de entrada  $\Sigma$ .
- 3) Un alfabeto de pila  $Z$ .
- 4) Un estado inicial  $q_0 \in Q$ .
- 5) Un símbolo de fondo de pila  $z_0 \in Z$ .
- 6) Un conjunto de estados finales  $F \subseteq Q$ .
- 7) Un conjunto de transiciones  $\delta: Q \times (\Sigma \cup \{\lambda\}) \times Z \rightarrow 2^{Q \times Z^*}$ .

### Observad que...

... en relación con  $Q$ ,  $\Sigma$ ,  $q_0$ , y  $F$  sin las dos  $Z$ , un autómata con pila coincide totalmente con la definición de *autómata finito*.

Si  $(q^1, \alpha) \in \delta(q, a, z)$ , lo notaremos  $(q, a, z) \vdash (q^1, \alpha)$ .

Una transición indica que, dado un estado, una letra del alfabeto  $\Sigma$  (o la palabra vacía, ya veremos por qué) y una cima de pila, es necesario evolucionar a otro estado y eliminar la cima de la pila para después añadir la secuencia de símbolos de pila.

### Observad que...

... en cada transición se puede desapilar un solo símbolo, pero apilar cero, uno o varios.

El símbolo de cima de pila  $z \in Z$  tiene dos funciones:

- 1) Indica cuál debe ser la cima de la pila para poder aplicar la transición.
- 2) Es el símbolo que hay que desapilar (si no queremos modificar la pila, deberemos volverlo a apilar).




La  $\alpha$  es la secuencia de símbolos de pila que debemos apilar además de pasar al estado  $q'$ .

La notación  $(q, a, z) \vdash (q', \alpha)$  significa intuitivamente que, cuando el autómata está en el estado  $q$  leyendo el símbolo  $a$  en la cinta de entrada y  $z$  es la cima de la pila, se puede desapilar  $z$ , después apilar  $\alpha$ , y mover el cabezal de la cinta una posición a la derecha y pasar al estado  $q'$ .

#### Nota

Si  $a$  es la palabra vacía ( $\lambda$ -transición), no hay cambios en la cinta de entrada, sólo en la pila.


El autómata con pila es por definición no determinista; en un cierto momento puede haber varias transiciones posibles. Si en algún autómata con pila en todo momento sólo se puede aplicar como mucho una sola transición, haremos constar que es determinista. 

Decimos que un autómata con pila es determinista si para todo estado  $q \in Q$  y para todo símbolo en la cima de la pila  $z \in Z$  se verifica:

1) Si  $\delta(q, \lambda, z) \neq \emptyset$ , entonces  $\delta(q, a, z) = \emptyset \forall a \in \Sigma$

2)  $|\delta(q, a, z)| \leq 1 \forall a \in \Sigma \cup \{\lambda\}$

Es preciso recordar que todo autómata finito no determinista tiene un equivalente determinista. No sucede lo mismo en los autómatas con pila, ya que no todos tienen un equivalente determinista; se pueden realizar más acciones con el no determinista.

 Recordad la construcción de subconjuntos, de que hemos tratado en el apartado 2 del módulo "Autómatas finitos y lenguajes regulares" de esta asignatura.

Como ya pasaba con la definición de una gramática, podemos expresar un autómata con pila dando sólo su conjunto de transiciones. El resto de los componentes ya aparece de forma implícita, excepto el conjunto de estados finales (para los cuales podemos establecer unos criterios de notación, por ejemplo  $q_f$ ,  $q_{f1}$ ,  $q_{f2}$ , ...).

Queda claro que la potencia de expresión del autómata con pila es como mínimo la del autómata finito, y si no usamos la pila, es exactamente la misma.

## 6.2. Descripción instantánea de un autómata con pila

Para describir la configuración o situación en un momento determinado de un autómata con pila, se define lo que se conoce como *descripción instantánea*. Incluye el estado en curso, el fragmento de cinta de entrada que queda por leer (a la derecha del cabezal) y el contenido de la pila.

#### Representación de autómatas con pila

Algunos autores proponen representar los autómatas con pila por medio de un diagrama de transiciones (representando con redondas los estados y con flechas las transiciones entre estados) como el de los autómatas finitos, pero añadiendo a la etiqueta de las flechas (carácter en la entrada que provoca la transición) el comportamiento de la pila (símbolo que se debe desapilar y cadena de símbolos que se tienen que apilar posteriormente).

Más formalmente, se puede definir de la forma que exponemos a continuación:

La **descripción instantánea** (o **configuración**) de un **autómata con pila** proporciona información sobre su estado global en un momento determinado. Está formada por tres componentes  $(q, m, \alpha)$ , donde:

- 1)  $q \in Q$  designa el estado actual.
- 2)  $m \in \Sigma^*$  designa la subpalabra de la entrada entre el símbolo que apunta al cabezal y el extremo derecho de la cinta de entrada.
- 3)  $\alpha \in Z^*$  designa el contenido de la pila (leído de arriba abajo).

Si tenemos la descripción instantánea  $(q, am, z\alpha)$  y la transición  $(q, a, z) \vdash (q', \beta)$ , el autómata con pila puede evolucionar hacia  $(q', m, \beta\alpha)$ . Más formalmente, tenemos que:

$$(q, am, z\alpha) \vdash^* (q', m, \beta\alpha),$$

y diremos que cuando el autómata está en el estado  $q$  y lee el símbolo  $a$ , evoluciona hacia el estado  $q'$ , desapila el símbolo  $z$  y apila la palabra  $\beta$ .

La **descripción instantánea inicial** para todo autómata con pila es  $(q_0, m, z_0)$ , donde  $q_0$  es el estado inicial del autómata,  $m$  es la palabra de la cinta de entrada y  $z_0$  es el símbolo de fondo de pila.


#### Observad que...

... utilizamos el símbolo  $\vdash$  tanto para indicar el paso entre dos descripciones instantáneas, como para indicar una imagen según la función de transición  $\delta$ .

### 6.3. Lenguaje aceptado por un autómata con pila

De manera parecida a lo que hemos hecho en las gramáticas con las derivaciones, si  $(q, m, \alpha) \vdash (q', m', \alpha')$ , diremos que hay una **transición directa** de una descripción instantánea  $(q, m, \alpha)$  a otra  $(q', m', \alpha')$ .

Utilizaremos la notación  $(q, m, \alpha) \vdash (q', m', \alpha')$  si en cero o más transiciones directas consecutivas se puede pasar de  $(q, m, \alpha)$  a  $(q', m', \alpha')$ .

Ahora ya estamos en condiciones de definir el lenguaje aceptado por un autómata con pila de la misma forma que lo hemos hecho para el lenguaje aceptado por un autómata finito. Sin embargo, en los autómatas con pila se suelen utilizar dos definiciones alternativas de aceptación: 

#### Otra definición

A veces también se considera una tercera definición de *aceptación*, que es la unión de las dos presentadas: la aceptación por pila vacía y estado final.

1) **Aceptación por pila vacía.** Con esta definición se aceptan las palabras que dejan la pila vacía después de ser consumidas.

**2) Aceptación por estado final.** Con esta definición (que es equivalente a la aceptación en autómatas finitos) se aceptan las palabras que llegan a un estado final después de ser consumidas.

No importa cuál de las dos se utilice, porque cada una puede simular a la otra, es decir, son equivalentes.

El **lenguaje aceptado por estado final** por un autómata con pila  $M$ , denotado  $L(M)$ , es el conjunto de las palabras que permiten pasar de la descripción instantánea inicial a una descripción instantánea con estado final después de haber consumido todos los símbolos de la cinta de entrada. Formalmente, lo expresamos así:

$$L(M) = \{m \in \Sigma^* \mid \exists q, \alpha (q_0, m, z_0) \xrightarrow{*} (q, \lambda, \alpha) \wedge q \in F\}.$$

El **lenguaje aceptado por pila vacía** por un autómata con pila  $M$ , denotado  $N(M)$ , es el conjunto de las palabras que permiten pasar de la descripción instantánea inicial a una descripción instantánea en la que la pila está vacía después de haber consumido todos los símbolos de la cinta de entrada. Formalmente, lo expresamos así:

$$N(M) = \{m \in \Sigma^* \mid \exists q (q_0, m, z_0) \xrightarrow{*} (q, \lambda, \lambda)\}.$$

Cuando utilizamos la aceptación por pila vacía, no tendrá sentido indicar cuál es el conjunto de estados finales; por tanto, consideraremos  $F = \emptyset$ .

## 6.4. Ejemplos

Como regla general, daremos el autómata con pila en su versión determinista siempre que sea posible; si no es posible, daremos la versión no determinista y lo justificaremos.

**Ejemplo 1.** Encontramos un autómata con pila para nuestro primer lenguaje incontextual,  $L = \{a^n b^n \mid n \geq 0\}$ .

1) En primer lugar damos un autómata con pila que acepta las palabras del lenguaje por pila vacía:

- $Q = \{q_0, q_1\}$ ,
- $\Sigma = \{a, b\}$ ,
- $Z = \{z_0, a\}$ ,

### Recordad que...

...  $q_0$  será siempre el estado inicial y que  $z_0$  será siempre el símbolo de fondo de pila. Por eso ya no los especificamos.

El conjunto de transiciones de este autómata es el siguiente:

$$\begin{aligned}(q_0, \lambda, z_0) &\vdash (q_0, \lambda), \\(q_0, a, z_0) &\vdash (q_0, az_0), \\(q_0, a, a) &\vdash (q_0, aa), \\(q_0, b, a) &\vdash (q_1, \lambda), \\(q_1, b, a) &\vdash (q_1, \lambda), \\(q_1, \lambda, z_0) &\vdash (q_1, \lambda).\end{aligned}$$

Si nos adentramos en este conjunto de transiciones, podremos ver que:


**a)** La primera transición sirve para aceptar la palabra vacía (que pertenece al lenguaje, cuando  $n = 0$ ). Simplemente, si no hay nada en la cinta de entrada y sólo tenemos el símbolo de fondo de pila, desapilamos este símbolo y dejamos la pila vacía.

**b)** Las transiciones segunda y tercera van consumiendo las  $a$  de la cinta de entrada y las van apilando (memorizándolas) para después poder comprobar que hay exactamente el mismo número de  $b$ . Para hacer todo esto no es necesario cambiar de estado.

**c)** La cuarta transición trata la aparición en la cinta de entrada de la primera  $b$  después de las  $a$ . Cambiamos de estado para indicar que ahora ya no pueden aparecer más  $a$ , y eliminamos una  $a$  de la pila (se desapila automáticamente como cima de la pila) para contrarrestar la  $b$  leída.

**d)** La quinta transición hace lo mismo que la cuarta, pero sin cambiar de estado: desapila una  $a$  por cada  $b$  leída de la cinta de entrada. De esta manera, sabremos que tenemos tantas  $b$  como  $a$  si cuando llegamos al final de la cinta de entrada ya no queda ninguna  $a$  en la pila.

**e)** La sexta y última transición vacía la pila (y, por tanto, acepta la palabra) si llegamos al fondo de la pila a la vez que al final de la cinta de entrada.

Es necesario prestar atención a las transiciones vacías ( $\lambda$ -transiciones) porque pueden indicar dos cosas diferentes: 

- Que no se mueve el cabezal de la cinta de entrada (no consumen ningún símbolo) aunque queden símbolos en la cinta.
- Que realmente hemos llegado al final de la cinta de entrada (se ha consumido toda la palabra).

En el primer caso, la transición se puede aplicar siempre que coincidan el estado y la cima de la pila, independientemente de la cinta de entrada, y esto provoca que el autómata sea no determinista si hay alguna transición con el mismo estado y cima de pila (sea cual sea el símbolo de la cinta de entrada que

#### Observad que...

... si sólo queremos añadir símbolos a la pila, debemos apilar la antigua cima de pila además de lo que ponemos, porque la cima de pila siempre se desapila.

lea). Esto es precisamente lo que pasa con las dos primeras transiciones de nuestro ejemplo. Por tanto, el autómata con pila que hemos dado es no determinista.

En cambio, con la sexta transición (también transición vacía) no introducimos no-determinismo, no hay ninguna otra regla con estado  $q_1$  y cima de pila  $z_0$ . Si consideramos el mismo lenguaje, pero sin aceptar la palabra vacía (es decir,  $L = \{a^n b^n \mid n \geq 1\}$ ), tendremos las siguientes transiciones, que corresponden a un autómata con pila determinista:

$$\begin{aligned}(q_0, a, z_0) &\vdash (q_0, az_0), \\(q_0, a, a) &\vdash (q_0, aa), \\(q_0, b, a) &\vdash (q_1, \lambda), \\(q_1, b, a) &\vdash (q_1, \lambda), \\(q_1, \lambda, z_0) &\vdash (q_1, \lambda).\end{aligned}$$

Fijaos en que en todo momento sólo se puede aplicar una transición.

Si en un momento determinado se duda del hecho de que el autómata pueda aceptar palabras que no pertenecen al lenguaje al aplicar la última transición antes de hora, es necesario recordar la definición de *aceptación por pila vacía*: se aceptan las palabras que acaban con pila vacía después de haber consumido todos los símbolos de la cinta de entrada.

#### Observación

Si en la cinta de entrada hay más  $b$  que  $a$ , cuando lleguemos al fondo de pila quedan  $b$  en la cinta de entrada y, si aplicamos la última transición, dejamos la pila vacía, pero no hemos consumido todos los símbolos en la entrada (no se acepta).

También podemos indicar el conjunto de transiciones en forma de tabla, de manera que cada fila corresponda a un estado y cada columna, a una combinación de símbolo de entrada (que será  $\lambda$  en el caso de las  $\lambda$ -transiciones) y símbolo de la cima de la pila. Para aquellas combinaciones que no tengan ninguna transición posible desde ningún estado, no será necesario que pongamos la columna correspondiente.

Así, el autómata anterior se puede representar como:

$\delta$	$a / z_0$	$a / a$	$b / a$	$\lambda / z_0$
$q_0$	$(q_0, az_0)$	$(q_0, aa)$	$(q_1, \lambda)$	$\emptyset$
$q_1$	$\emptyset$	$\emptyset$	$(q_1, \lambda)$	$(q_1, \lambda)$

2) En segundo lugar damos un autómata con pila que acepta las siguientes palabras del lenguaje por estado final; en este caso, aceptaremos también la palabra vacía:

- $Q = \{q_0, q_1, q_f\}$ ,
- $\Sigma = \{a, b\}$ ,
- $Z = \{z_0, a\}$ ,
- $F = \{q_f\}$ ,

Siguiendo con la notación propuesta, puesto que sólo hay un estado final, lo denominamos  $q_f$ .

y el conjunto de transiciones es el siguiente:

$$\begin{aligned}(q_0, \lambda, z_0) &\vdash (q_f, \lambda), \\(q_0, a, z_0) &\vdash (q_0, az_0), \\(q_0, a, a) &\vdash (q_0, aa), \\(q_0, b, a) &\vdash (q_1, \lambda),\end{aligned}$$

$$\begin{aligned}(q_1, b, a) &\vdash (q_1, \lambda), \\(q_1, \lambda, z_0) &\vdash (q_f, \lambda).\end{aligned}$$

Podemos representar este conjunto en forma de tabla:

$\delta$	$a / z_0$	$a / a$	$b / a$	$\lambda / z_0$
$q_0$	$(q_0, az_0)$	$(q_0, aa)$	$(q_1, \lambda)$	$(q_f, \lambda)$
$q_1$	$\emptyset$	$\emptyset$	$(q_1, \lambda)$	$(q_1, \lambda)$

**Ejemplo 2.** Encontramos un autómata con pila para el lenguaje de las cadenas de paréntesis balanceados.

1) En primer lugar damos un autómata con pila que acepta las palabras del lenguaje por pila vacía, cuyo sistema de transiciones es el siguiente:

$$\begin{aligned}(q_0, [, z_0) &\vdash (q_0, [z_0), \\(q_0, [, D) &\vdash (q_0, [D), \\(q_0, ], D) &\vdash (q_0, \lambda), \\(q_0, \lambda, z_0) &\vdash (q_0, \lambda).\end{aligned}$$

#### Nota

A partir de ahora sólo daremos el conjunto de transiciones; el resto de los componentes queda implícito.

Podemos representar este conjunto en forma de tabla:

$\delta$	$[ / z_0$	$[ / [$	$] / [$	$\lambda / z_0$
$q_0$	$(q_0, [z_0)$	$(q_0, [D)$	$(q_0, \lambda)$	$(q_f, \lambda)$

Observad que en este caso debemos utilizar un solo estado porque los paréntesis cuadrados de abrir pueden aparecer después de que hayan salido los de cerrar (en el ejemplo anterior tenían que salir primero todas las  $a$  y después todas las  $b$ ).

Aún sacamos más partido de la estructura de pila: aparte de “contar” los paréntesis de abrir\*, los guarda en el orden en que han aparecido (apilándolos, en las dos primeras reglas) y así puede comprobar que cada uno tiene su correspondiente de cerrar (desapilándolos si aparece la pareja, en la tercera regla). La cuarta regla nos permite vaciar la pila cuando se acaban los símbolos de la cinta de entrada.

\* Para después poder asegurar que hay tantos como de cerrar.

El funcionamiento del autómata es el siguiente:

- Siempre que aparece un paréntesis de abrir se apila (junto con la cima de la pila en curso que se había desapilado).
- Cuando aparece un paréntesis de cerrar, se desapila su par de abrir (si la cima de la pila no es un paréntesis de abrir, el autómata se bloquea porque no hay ninguna transición que lo prevea y la palabra no está balanceada).
- Finalmente, cuando se llega al final de la entrada con el símbolo de fondo de pila en la cima, se desapila y se acepta la palabra.



La tabla de transiciones correspondientes es la siguiente:

$\delta$	$[ / z_0$	$[ / [$	$] / [$	$\lambda / z_0$
$q_0$	$(q_0, [z_0)$	$(q_0, [[$	$(q_0, \lambda)$	$(q_f, \lambda)$

**Ejemplo 3.** Encontramos un autómata con pila para el lenguaje incontextual  $L = \{ww^R \mid w \in (a + b)^*\}$ .

Recordad que hemos definido  $w^R$  como la inversión de una palabra, la palabra escrita al revés, en el subapartado 3.3. del módulo "Alfabetos, palabras y lenguajes" de esta asignatura.

En este ejemplo veremos un no-determinismo más claro, más parecido al de los autómatas finitos (no asociado a las transiciones vacías, como en los ejemplos anteriores).

En la mayoría de las ocasiones es conveniente hacerse un guión previo (una especie de programa) de lo que deben hacer las transiciones, y a partir de éste, ir pensando las transiciones concretas que cubren cada aspecto. En cierto modo, se trata de una descripción intuitiva de lo que debe hacer el autómata con pila.

En este sentido, también puede servir de ayuda escribir un diagrama de transiciones\* con una indicación adicional en las transiciones del comportamiento de la pila\*\*.

\* Como el de los autómatas finitos.

\*\* Es decir, qué apilamos y qué desapilamos.


En nuestro enunciado observamos que debemos guardar (apilar) la primera mitad de la palabra de entrada para comprobar después que lo que queda de la palabra es efectivamente su inverso. ¿Cómo sabemos dónde está la mitad de la palabra? Aquí es donde interviene el no-determinismo, y el autómata tiene que poder decidir en un determinado momento entre dos caminos: continuar guardando la palabra o comenzar la comprobación. Esto da lugar a una serie de alternativas en las secuencias de transiciones, pero mientras una sea la buena, ya es suficiente. Aquí radica la potencia del no-determinismo.

Un guión o esquema de funcionamiento sería el siguiente:

```

mientras no mitad_de_la_palabra hacer
  leer símbolo de entrada y apilarlo
fmientras;
mientras hay_simbolos_entrada hacer
  si símbolo_entrada = cima_pila entonces
    desapilarlo
  sino bloqueo
  fsi
fmientras
si simbolo_fondo_pila = cima_pila entonces
  aceptar palabra
fsi

```

Toda operación no determinista se traduce en una o varias transiciones con la misma parte izquierda. 



El autómata con pila de aceptación por pila vacía y estado final (queda más claro) correspondiente al guión es:

$$\begin{aligned}
 (q_0, \lambda, z_0) &\vdash (q_f, \lambda), \\
 (q_0, a, z_0) &\vdash (q_0, az_0), \\
 (q_0, b, z_0) &\vdash (q_0, bz_0), \\
 (\mathbf{q_0}, \mathbf{a}, \mathbf{a}) &\vdash (\mathbf{q_0}, \mathbf{aa}), \\
 (q_0, b, a) &\vdash (q_0, ba), \\
 (q_0, a, b) &\vdash (q_0, ab), \\
 (\underline{q_0}, \underline{b}, \underline{b}) &\vdash (\underline{q_0}, \underline{bb}), \\
 (\mathbf{q_0}, \mathbf{a}, \mathbf{a}) &\vdash (\mathbf{q_1}, \lambda), \\
 (\underline{q_0}, \underline{b}, \underline{b}) &\vdash (\underline{q_1}, \lambda), \\
 (q_1, a, a) &\vdash (q_1, \lambda), \\
 (q_1, b, b) &\vdash (q_1, \lambda), \\
 (q_1, \lambda, z_0) &\vdash (q_f, \lambda).
 \end{aligned}$$

La tabla de transiciones correspondientes es la siguiente:

$\delta$	$\lambda / z_0$	$a / z_0$	$b / z_0$	$a / a$	$b / a$	$a / b$	$b / b$
$q_0$	$(q_f, \lambda)$	$(q_0, az_0)$	$(q_0, bz_0)$	$(q_0, aa)$ $(q_1, \lambda)$	$(q_0, ba)$	$(q_0, ab)$	$(q_0, bb)$ $(q_1, \lambda)$
$q_1$	$(q_f, \lambda)$	$\emptyset$	$\emptyset$	$(q_1, \lambda)$	$\emptyset$	$\emptyset$	$(q_1, \lambda)$

Por tanto, el funcionamiento de este autómata con pila es el siguiente:

- a) La primera transición acepta la palabra vacía.
- b) Las transiciones de la segunda a la séptima corresponden al primer bucle del guión: consumimos la primera mitad de la palabra ( $w$ ) apilándola.
- c) Las transiciones de la octava a la undécima corresponden al segundo bucle: comprobamos que lo que queda en la cinta es la palabra inversa ( $w^R$ ) de la palabra que hemos guardado en la pila (la primera de la entrada coincide con la última de la palabra en la pila, que es precisamente la cima de la pila) y, mientras coincide, la desapilamos.
- d) Finalmente, la duodécima transición pasa al estado final y vacía la pila en caso de que se acaben los símbolos de la cinta de entrada cuando encontramos el símbolo de fondo de pila en la pila. Entonces, se acepta la palabra.

El no-determinismo queda expresado en dos pares de reglas, las que hemos marcado en negrita y las que hemos subrayado. Si os fijáis, en los dos casos tenemos:


- La misma parte izquierda: estado  $q_0$ ,  $a$  o  $b$  en la cinta de entrada y el mismo en la cima de la pila.
- Distinta parte derecha: una no cambia de estado y apila el símbolo leído, y la otra cambia al estado  $q_1$  desapilando la palabra que coincide en la entrada y en la cima de la pila.

Una alternativa corresponde al hecho de continuar almacenando la palabra en la pila (esto supone que no ha llegado a la mitad de la palabra) y la otra corresponde al hecho de comenzar a comprobar que la segunda mitad (aún por consumir) es el inverso de la primera mitad (en la pila).

Es fácil convencerse de que este comportamiento es imposible de realizar de forma determinista.

## 6.5. Equivalencia entre autómatas con pila y gramáticas incontextuales

Hasta ahora hemos presentado por separado las gramáticas incontextuales (gramática generativa con unas reglas con ciertas restricciones) y los autómatas con pila (autómata finito con  $\lambda$ -transiciones y una memoria con estructura de pila). Hemos utilizado como ejemplos los mismos lenguajes, pero aún no hemos dicho nada, en general, de la relación que se establece entre las gramáticas incontextuales y los autómatas con pila.


En este último apartado demostraremos (informalmente) que los autómatas con pila\* y las gramáticas incontextuales son equivalentes en poder expresivo. 

\* Por defecto nos referimos a autómatas con pila no deterministas.

Los lenguajes aceptados por un autómata con pila (no determinista) son exactamente los lenguajes incontextuales (generados por una gramática incontextual).

Dicho de otra manera, un lenguaje es incontextual si, y sólo si, es aceptado por algún autómata con pila: todo lenguaje incontextual tiene un autómata con pila que lo reconoce, y todo lenguaje aceptado por algún autómata con pila es un lenguaje incontextual.

Así, nos encontramos con una clase de lenguajes caracterizada de forma parecida a la que hemos visto para los lenguajes regulares. En lugar de los lenguajes regulares, tenemos los lenguajes incontextuales, y en lugar de los autómatas finitos, tenemos los autómatas con pila.

Sin embargo, hay un par de diferencias importantes entre los autómatas finitos y los autómatas con pila: 

1) Todo autómata finito no determinista se puede determinizar (con el mismo poder expresivo) y, en cambio, como hemos visto, no todo autómata con pila no determinista tiene su equivalente determinista. Por tanto, los autómatas con pila deterministas sólo caracterizan un subconjunto estricto de lenguajes incontextuales.

2) Todos los autómatas finitos equivalentes (es decir, autómatas finitos que reconocen el mismo lenguaje) tienen el mismo autómata mínimo, pero no hay una propiedad parecida en los autómatas con pila.

Volvemos a la equivalencia entre autómatas con pila y gramáticas incontextuales. No es nuestra intención hacer aquí una demostración formal de esta equivalencia, sino dar una idea intuitiva que nos convenza de este hecho.

En primer lugar, daremos una idea de cómo se construiría un autómata con pila a partir de una gramática incontextual y, en segundo lugar, veremos cómo se simularía un autómata con pila con una gramática incontextual. !

### 1) Construcción de un autómata con pila a partir de una gramática incontextual

Tal como hemos visto en formas normales, podemos suponer que todas las reglas de la gramática  $G$  adoptan la siguiente forma:

$$A \rightarrow aB_1B_2 \dots B_k,$$

donde  $a \in \Sigma \cup \{\lambda\}$  y  $k \geq 0^*$ .

El autómata con pila (de aceptación por pila vacía) equivalente tendrá los siguientes componentes:

- $Q = \{q\}$ , un solo estado  $q$ .
- $\Sigma$ , el conjunto de símbolos terminales de  $G$ , es el alfabeto de entrada.
- $V$ , el conjunto de variables de  $G$ , es el alfabeto de pila.
- $q$  es el estado inicial.
- $S$ , la variable inicial de  $G$ , es el símbolo de fondo de pila.
- El conjunto de transiciones se define de la manera siguiente: para cada regla  $A \rightarrow aB_1B_2 \dots B_k$  hay una transición  $(q, a, A) \vdash (q, B_1B_2 \dots B_k)$ .

Recordemos lo que “hace” esta transición: cuando en el estado  $q$  leemos el símbolo  $a$  en la cinta de entrada (o nada si  $a$  es la palabra vacía) y  $A$  está en la cima de la pila, desapilamos  $A$ , apilamos  $B_1B_2 \dots B_k$  ( $B_1$  será la nueva cima), y nos quedamos en el estado  $q$ .

Comenzando con  $S$  en el fondo de la fila, la primera transición aplicable será la correspondiente a una regla con  $S$  en la parte izquierda y con el símbolo terminal leído como primera posición de la parte derecha, y su efecto será apilar las variables que hay detrás después de desapilar  $S$ . A continuación, se podrá aplicar una transición correspondiente a una regla con  $B_1$  como parte izquierda y el siguiente símbolo terminal, y así sucesivamente. Las reglas como  $a = \lambda$

#### Nota

La demostración formal de la equivalencia entre autómatas con pila y gramáticas incontextuales se puede encontrar de forma muy detallada en la bibliografía.

\* Ésta es una variante de la forma normal de Greibach, que acepta la palabra vacía.

se convertirán en  $\lambda$ -transiciones, y no consumirán ningún símbolo de la cinta de entrada. Toda palabra consumida que deje la pila vacía será aceptada.

### Ejemplo de construcción de un autómata con pila a partir de una gramática incontextual

Por medio de este método encontramos un autómata con pila para el lenguaje de las cadenas no nulas de paréntesis balanceados.

Recordad que ya hemos dado una gramática en forma normal de Greibach que generaba este lenguaje, y que era la siguiente:

$$\begin{aligned} S &\rightarrow [B \mid [SB \mid [BS \mid [SBS, \\ B &\rightarrow ]. \end{aligned}$$

Según el procedimiento descrito, el autómata con pila tendrá el siguiente conjunto de transiciones (en el mismo orden que las reglas):

$$\begin{aligned} (q, [, S) &\vdash (q, B), \\ (q, [, S) &\vdash (q, SB), \\ (q, [, S) &\vdash (q, BS), \\ (q, [, S) &\vdash (q, SBS), \\ (q, ], B) &\vdash (q, \lambda). \end{aligned}$$

#### Recomendación

Realizad algún seguimiento (secuencia de descripciones instantáneas) para comprobar que este autómata con pila realmente acepta cadenas de paréntesis balanceados.

## 2) Simulación de un autómata con pila con una gramática incontextual

Acabamos de ver que todo lenguaje incontextual es aceptado por algún autómata con pila. Ahora nos queda por ver que todo autómata con pila acepta sólo lenguajes incontextuales, y entonces habremos demostrado que los autómatas con pila y las gramáticas incontextuales son equivalentes en poder expresivo.

Esto puede hacerse en los pasos que exponemos a continuación:

- a) Todo autómata con pila puede ser simulado con un autómata con pila con un solo estado.
- b) Todo autómata con pila con un estado tiene una gramática incontextual equivalente.

El segundo paso es fácil, sólo se trata de aplicar al revés la construcción del punto anterior: cada transición del autómata con pila da lugar a una regla del tipo  $A \rightarrow aB_1B_2 \dots B_k$ .

En cuanto al primer paso, se trata de simular un autómata con pila arbitrario con un autómata con pila de un solo estado. La idea es guardar toda la información de los estados en la pila. Los cambios en el conjunto de transiciones serán: cada transición  $(p, a, A) \vdash (q_0, B_1B_2 \dots B_k)$  la sustituimos por  $(q, a, \langle p A q_k \rangle) \vdash (q, \langle q_0 B_1 q_1 \rangle \langle q_1 B_2 q_2 \rangle \dots \langle q_{k-1} B_k q_k \rangle)$ , para todos los posibles valores de  $q_k$ .

#### Por ejemplo,...

... en el caso de  $k = 0$  nos dará una sola transición  $(q, a, \langle p A q_0 \rangle) \vdash (q, \lambda)$ , en el caso de  $k = 1$  nos dará dos transiciones, y así sucesivamente.

Intuitivamente, el autómata con un solo estado ( $q$ ) simula un autómata de pila conjeturando de forma no determinista qué estados adquirirá el autómata en ciertos momentos futuros, y guarda estas conjeturas en la pila, para verificar más tarde cuáles de éstas son correctas.

## 7. Propiedades de los lenguajes incontextuales

Una vez caracterizada la clase de los lenguajes incontextuales (los lenguajes de tipo 2, según la jerarquía de Chomsky) mediante las gramáticas incontextuales y los autómatas con pila (que hemos demostrado que son equivalentes), sólo nos queda por ver cuáles son las propiedades que cumplen.

Hay un lema del bombeo para los lenguajes incontextuales, de la misma manera que había uno para los lenguajes regulares. En su versión negada, este lema nos sirve para demostrar que un lenguaje no es incontextual (acabamos, como en los lenguajes regulares, mostrando las limitaciones de la clase y justificando la existencia de una clase superior).

En relación con el lema del bombeo para lenguajes regulares, consultad el apartado 6 del módulo "Autómatas finitos y lenguajes regulares" de esta asignatura.

A continuación veremos el comportamiento de los lenguajes incontextuales respecto a las operaciones básicas sobre lenguajes (unión, intersección, concatenación, complementario, etc.) y cómo podemos aprovechar esto para deducir las gramáticas incontextuales que los generan.

Finalmente, veremos algoritmos de decisión para resolver ciertos problemas que se pueden plantear sobre lenguajes incontextuales.

### 7.1. Lema del bombeo


Hay un lema del bombeo para lenguajes incontextuales similar al que tenemos para lenguajes regulares. De hecho, este lema expresa una propiedad que cumplen todos los lenguajes incontextuales y que en su forma contrapositiva nos permite demostrar que un cierto lenguaje no es incontextual.

Su planteamiento formal es el que presentamos a continuación:

Para todo lenguaje incontextual  $L$  hay una constante  $N \geq 0$  (que sólo depende de  $L$ ) tal que, si  $m \in L$  y  $|m| \geq N$ , entonces podemos dividirlo en cinco subpalabras,  $m = uvwxy$ , tal que:

- a)  $|v| \neq 0$ .
- b)  $|vwx| \leq N$ .
- c) Para todo  $i \geq 0$ ,  $uv^iwx^iy \in L$ .

Informalmente, para todo lenguaje incontextual  $L$ , toda palabra suficientemente larga de  $L$  se puede dividir en cinco factores tales que los tres factores del medio no son demasiado largos, el segundo y el cuarto factores no son simul-


taneamente nulos, y que es igual las veces que dupliquemos simultáneamente el segundo y el cuarto factor porque la palabra continuará perteneciendo al lenguaje. 

Al igual que sucedía con el lema del bombeo para los lenguajes regulares, este lema para los lenguajes incontextuales se utiliza sobre todo en su forma contrarrecíproca. Para llegar a la conclusión de que un lenguaje  $L$  no es incontextual, es suficiente con demostrar la siguiente propiedad:

Para todo  $N \geq 0$ , hay una palabra  $m \in L$  con  $|m| \geq N$  tal que, para todas las posibles maneras de dividir  $m$  en cinco subpalabras  $m = uvwxy$  con  $|vx| > 0$  y  $|vwx| \leq N$ , existe un  $i \geq 0$  tal que  $uv^iwx^iy \notin L$ .

#### Observad que...

... la diferencia más importante con el lema del bombeo para los lenguajes regulares es que “abombamos” simultáneamente dos subpalabras  $v$  y  $x$  separadas por una subpalabra  $w$ .

En la práctica, para demostrar que un lenguaje no es incontextual, debemos actuar de la siguiente manera: en primer lugar, tenemos que escoger una palabra general (en función de  $N$ ) que pertenezca al lenguaje y sea de longitud superior o igual a  $N$ , y después tenemos que demostrar que todas las posibles factorizaciones (o subdivisiones) que cumplen las condiciones indicadas (**a** y **b**) tienen un bombeo (un valor de  $i$ ) que provoca que  $uv^iwx^iy$  (la palabra con los bombeos correspondientes) no pertenezca al lenguaje. 

Pasemos a verlo con la ayuda de un ejemplo.

Utilizamos el lema del bombeo para demostrar que el lenguaje  $L = \{a^n b^n c^n \mid n \geq 0\}$  no es incontextual.

Tomamos la palabra  $m = a^N b^N c^N$  (observad que está en función de  $N$ , para cada  $N$  indica una palabra diferente y siempre de longitud superior a  $N$ ).

Consideremos ahora las factorizaciones  $m = uvwxy$  que cumplen las condiciones:

- a)  $|vx| > 0$ .
- b)  $|vwx| \leq N$ .

Por **b** sabemos que  $vwx$  no puede contener a la vez símbolos  $a$ ,  $b$  y  $c$  (están separadas por  $N$   $b$ ); y por **a** sabemos que alguno de los factores que hay que abombar,  $v$  o  $x$ , es no nulo. Las factorizaciones posibles (y su estudio) se pueden agrupar en cinco casos:

- 1)  $v$  y  $x$  sólo tienen  $a$ : entonces,  $uv^iwx^iy$  para  $i = 0$  ( $uwy$ ) tiene tantas  $b$  como  $c$  (todas en  $y$ , no afectadas), pero como mínimo, una  $a$  menos y, por tanto, no pertenece al lenguaje.
- 2)  $v$  y  $x$  sólo tienen  $b$ : entonces,  $uv^iwx^iy$  para  $i = 0$  ( $uwy$ ) tiene tantas  $a$  como  $c$  (todas en  $u$  y  $y$ , respectivamente, no afectadas), pero como mínimo una  $b$  menos y, por tanto, no pertenece al lenguaje.

3)  $v$  y  $x$  sólo tienen  $c$ : entonces,  $uv^iwx^iy$  para  $i = 0$  ( $uwy$ ) tiene tantas  $a$  como  $b$  (todas en  $u$ , no afectadas), pero como mínimo una  $c$  menos y, por tanto, no pertenece al lenguaje.

4)  $v$  y  $x$  tienen  $a$  y  $b$ : entonces,  $uv^iwx^iy$  para  $i = 0$  ( $uwy$ ) tiene más  $c$  (todas en  $y$ , no afectadas) que  $a$  o  $b$ , y por tanto, no pertenece al lenguaje.

5)  $v$  y  $x$  tienen  $b$  y  $c$ : entonces,  $uv^iwx^iy$  para  $i = 0$  ( $uwy$ ) tiene más  $a$  (todas en  $u$ , no afectadas) que  $b$  o  $c$ , y por tanto, no pertenece al lenguaje.

Después de estudiar todas las posibles factorizaciones que cumplen las condiciones impuestas por el lema y encontrar un valor para  $i$ , y que haga que la palabra  $uv^iwx^iy$  no pertenezca al lenguaje, podemos concluir que este lenguaje no es incontextual.


De la misma forma, podemos utilizar el lema del bombeo para demostrar que otros lenguajes parecidos no son incontextuales. Pero por desgracia, hay ciertos lenguajes que no son incontextuales, y el lema del bombeo no permite demostrarlo. Para estos casos hay una versión más fuerte del lema del bombeo, el conocido como **lema de Ogden** y que definimos a continuación:

Para todo lenguaje incontextual  $L$ , hay una constante  $N \geq 0$  tal que, si  $m \in L$  y si marcamos cualquier  $N$  o más posiciones de  $m$  como “distinguidas”, entonces podemos dividir la palabra en cinco subpalabras,  $m = uvwxy$ , tal que:

- 1) Las subpalabras  $v$  y  $x$  tienen como mínimo una posición distinguida.
- 2) La subpalabra  $vw$  tiene como máximo  $N$  posiciones distinguidas.
- 3) Para todo  $i \geq 0$ ,  $uv^iwx^iy \in L$ .

Observad que el lema del bombeo para lenguajes incontextuales es un caso especial del lema de Ogden, en el que todas las posiciones son distinguidas.

Ésta es la forma habitual de demostrar la incontextualidad de un lenguaje más o menos formalmente. Sin embargo, a veces sólo nos interesa ver intuitivamente (y de forma rápida) si un lenguaje es incontextual o no. En estos casos puede ser útil razonar sobre su posible autómata con pila.

Si un lenguaje es incontextual, debe haber un autómata con pila que lo reconozca. Si llegamos a la conclusión de que no es posible construir este autómata con pila, podremos decir que el lenguaje no es incontextual. 

En el lenguaje del ejemplo anterior debemos comprobar que el número de  $a$ ,  $b$  y  $c$  es el mismo. Si al consumir las  $a$  las apilamos, para comprobar si hay el mismo número de  $b$  las tenemos que desapilar (por cada  $b$  consumi-

#### Nota

En la bibliografía podéis encontrar ejemplos de lenguajes incontextuales, a los que no se puede aplicar el lema del bombeo pero sí el de Ogden.



da desapilamos una  $a$ ) y llegar al fondo de la pila con todas las  $b$  consumidas. Cuando llega el momento de comprobar que también hay el mismo número de  $c$ , no tenemos nada para hacerlo. No es posible construir un autómata con pila para este lenguaje, porque es necesaria más “memoria”.

## 7.2. Propiedades de cierre

Pasemos a ver el comportamiento de la familia de lenguajes incontextuales respecto a las operaciones básicas sobre lenguajes: operaciones conjuntistas (unión, intersección, complementación y diferencia), concatenación, inversión y estrella de Kleene.

Observad las definiciones de operaciones sobre lenguajes en el módulo “Alfabetos, palabras y lenguajes” de esta asignatura.

Nos interesará saber tanto qué operaciones conservan la incontextualidad de los lenguajes (dados dos lenguajes incontextuales, el formado por la aplicación de la operación sobre éstos también lo es) como cuáles no la conservan (en qué casos no podemos estar seguros de que la operación aplicada sobre uno o dos lenguajes incontextuales nos da un lenguaje incontextual).

El interés de este comportamiento es triple: !

- 1) Nos puede ayudar a construir gramáticas para lenguajes incontextuales, ya que permite descomponer un lenguaje complejo en lenguajes más simples relacionados por las operaciones básicas (más simples desde el punto de vista de encontrar una gramática que los genere).
- 2) Nos puede ayudar a demostrar que un cierto lenguaje es incontextual, por medio de la misma descomposición.
- 3) Nos puede ayudar a demostrar que ciertos lenguajes no son incontextuales, construyendo a partir de estos lenguajes que sabemos positivamente que no son incontextuales, y utilizar sólo operaciones que conserven los lenguajes incontextuales.

Cuando al aplicar la operación sobre lenguajes incontextuales obtenemos siempre un lenguaje incontextual, diremos que éstos son **lenguajes incontextuales cerrados respecto de la operación**. Si no podemos asegurar que lo sean siempre (a pesar de que lo sean en casos particulares), diremos que no son cerrados respecto a la operación. !

Comenzaremos tomando en consideración las operaciones que conservan la incontextualidad.

Los lenguajes incontextuales son **cerrados respecto a la unión**. Dicho de otra manera, la unión de dos lenguajes incontextuales es un lenguaje incontextual.

Esta propiedad nos puede ayudar a encontrar una gramática que genere lenguajes incontextuales que se puedan descomponer en la unión de lenguajes, como el del ejemplo siguiente, en el que queremos construir una gramática que genere el lenguaje  $L = \{a^i b^j \mid i \neq j\}$ .

Apreciamos fácilmente que nuestro lenguaje se puede expresar como la unión de dos lenguajes:

$$L = \{a^i b^j \mid i \neq j\} = \{a^i b^j \mid i < j \vee i > j\} = \{a^i b^j \mid i < j\} \cup \{a^i b^j \mid i > j\},$$

Una posible gramática para cada uno de los lenguajes es:

$$A \rightarrow a \mid aA \mid aAb \text{ y } B \rightarrow b \mid Bb \mid aBb.$$

Si incorporamos la regla que nos permite escoger entre un lenguaje u otro,  $S \rightarrow A \mid B$ , obtenemos la gramática para el lenguaje inicial que es la unión de los otros dos; esta gramática es la siguiente:

$$\begin{aligned} S &\rightarrow A \mid B, \\ A &\rightarrow a \mid aA \mid aAb, \\ B &\rightarrow b \mid Bb \mid aBb. \end{aligned}$$

Los lenguajes incontextuales son **cerrados respecto a la concatenación**. Dicho de otro modo, la concatenación de dos lenguajes incontextuales es un lenguaje incontextual.

Esta propiedad nos puede ayudar a encontrar una gramática que genere lenguajes incontextuales expresables como concatenación de lenguajes, como el caso del ejemplo siguiente, en el que se debe construir una gramática que genere el lenguaje  $L = \{a^i b^j c^j d^i \mid i, j \geq 1\}$ .

Nuestro lenguaje se puede expresar como la concatenación de dos lenguajes:

$$L = \{a^i b^j c^j d^i \mid i, j \geq 1\} = \{a^i b^j \mid i \geq 1\} \cdot \{c^j d^j \mid j \geq 1\},$$

Una posible gramática para cada uno de los lenguajes es:

$$A \rightarrow ab \mid aAb \text{ y } B \rightarrow cd \mid cBd$$

Si incorporamos la regla que nos permite concatenar estos dos lenguajes,  $S \rightarrow AB$ , obtenemos la gramática para el lenguaje inicial que es la concatenación de los otros dos:

$$\begin{aligned} S &\rightarrow AB, \\ A &\rightarrow ab \mid aAb, \\ B &\rightarrow cd \mid cBd. \end{aligned}$$

#### Observad que...

... los conjuntos de variables utilizadas en las dos gramáticas son disjuntos.

#### Observad que...

... los conjuntos de variables utilizadas en las dos gramáticas son disjuntos.

Una vez que ya sabemos que son cerrados respecto a la concatenación, es fácil concluir que los lenguajes incontextuales son **cerrados respecto a la estrella de Kleene (\*)** y el **cierre positivo (+)**. Dicho de otra forma, la estrella de Kleene (y el cierre positivo) de un lenguaje incontextual es un lenguaje incontextual.

Si tenemos una gramática (con símbolo inicial  $S'$ ) que genera un lenguaje incontextual  $L$ , las gramáticas que generan los lenguajes también incontextuales  $L^*$  y  $L^+$  son, respectivamente:

$$S \rightarrow \lambda \mid S'S \text{ y } S \rightarrow S' \mid S'S.$$

Finalmente, también se mantiene la incontextualidad en el lenguaje inverso.

Los lenguajes incontextuales son **cerrados respecto a la inversión**. Dicho de otra manera, la inversión de un lenguaje incontextual es un lenguaje incontextual.

Si tenemos una gramática que genera un lenguaje incontextual  $L$ , la gramática que genera el lenguaje inverso  $L^R$  es la misma que genera  $L$ , pero invirtiendo las partes derechas de todas las reglas.

Ahora le toca el turno a las operaciones sobre lenguajes que no conservan la incontextualidad: la intersección y la complementación.

Los lenguajes incontextuales no son cerrados respecto a la intersección. Dicho de otra manera, la intersección de dos lenguajes incontextuales no siempre es un lenguaje incontextual.

Veámoslo con un ejemplo.

Consideremos los lenguajes  $L_1 = \{a^i b^j c^j \mid i, j \geq 1\}$  y  $L_2 = \{a^i b^j c^j \mid i, j \geq 1\}$ , ambos incontextuales y con las siguientes gramáticas que los generan:

$$\begin{aligned} S &\rightarrow AB, \\ A &\rightarrow ab \mid aAb, \\ B &\rightarrow c \mid cB, \end{aligned}$$

$$\begin{aligned} S' &\rightarrow CD, \\ C &\rightarrow aC \mid a, \\ D &\rightarrow bc \mid bDc. \end{aligned}$$

Sin embargo, la intersección de los dos es un lenguaje que se ha demostrado que no es incontextual:

$$L = \{a^i b^j c^j \mid i, j \geq 1\} \cap \{a^i b^j c^j \mid i, j \geq 1\} = \{a^i b^i c^i \mid i \geq 1\}.$$

Está claro que esto no nos puede servir para demostrar que un lenguaje es incontextual o para construir una gramática incontextual. Como alternativa nos puede servir otra propiedad en la que interviene la intersección, pero con un lenguaje regular, la cual expondremos a continuación:

La intersección de un lenguaje incontextual y un lenguaje regular es un lenguaje incontextual. Dicho de otra manera, los lenguajes incontextuales son cerrados respecto a la intersección con un lenguaje regular.

Esta propiedad también se puede utilizar, como ya hemos avanzado, para simplificar la demostración de que un cierto lenguaje no es incontextual.

Por ejemplo, para demostrar que el lenguaje  $L = \{w \in (a + b + c)^* \mid |w|_a = |w|_b = |w|_c\}$  no es incontextual, lo podemos interseccionar con el lenguaje regular  $a^* b^* c^*$  y obtenemos un lenguaje no incontextual ya conocido, que es el que vemos a continuación:

$$L \cap a^* b^* c^* = \{a^n b^n c^n \mid n \geq 0\}.$$

Puesto que el lenguaje resultante no es incontextual, llegamos a la conclusión (gracias a la propiedad que acabamos de enunciar) de que  $L$  tampoco lo es (si lo fuera, la intersección con un lenguaje regular debería ser incontextual).

Finalmente, la última propiedad que nos queda por enunciar es la que exponemos a continuación:

Los lenguajes incontextuales no son cerrados respecto a la complementación. Dicho de otra manera, el complementario de un lenguaje incontextual no siempre es un lenguaje incontextual.

La explicación es que, dados dos lenguajes incontextuales  $L_1$  i  $L_2$ , su intersección se puede expresar de la manera siguiente:

$$L_1 \cap L_2 = ((L_1 \cap L_2)^c)^c = (L_1^c \cup L_2^c)^c.$$

Consultad la aplicación del lema del bombeo en el subapartado 7.1. de este módulo didáctico.



Entonces, si el complementario de un lenguaje incontextual fuera siempre incontextual, tendríamos que la intersección de dos lenguajes incontextuales también debería ser incontextual, ya que la unión es cerrada respecto a la incontextualidad. Pero, ya hemos visto que la intersección de dos lenguajes incontextuales no siempre es un lenguaje incontextual.

Por ejemplo, el lenguaje  $L = \{ww \mid w \in (a+b)^*\}$  no es incontextual y, en cambio, su complementario sí lo es.


### Actividad

7.1. Encontrad una gramática para el complementario de este lenguaje.


## 7.3. Algoritmos de decisión

En este último apartado distinguiremos las preguntas sobre lenguajes incontextuales a las que siempre podemos responder (hay un algoritmo que las resuelve) de aquéllas a las que, en general, no podemos responder (no hay ningún algoritmo que las resuelva).

Comencemos por algunas de las cuestiones a las que sí podemos responder.

¿Hay un algoritmo para determinar si un lenguaje incontextual está vacío, es finito o infinito? 


De hecho, implícitamente ya hemos visto métodos en los apartados anteriores para responder a estas preguntas. Si partimos de la gramática que genera el lenguaje incontextual correspondiente, podemos decir cuándo un lenguaje incontextual es vacío.

 Consultad en el apartado 4.2. de este módulo didáctico el método para eliminar símbolos inútiles y en el subapartado 5.1. cómo se puede pasar una gramática a forma normal de Chomsky.

Si la variable inicial  $S$  es coaccesible (por definición, si de ésta se puede derivar alguna palabra), el lenguaje generado por la gramática no es vacío (y si no es coaccesible, es vacío).

Si partimos de la gramática en forma normal de Chomsky y sin símbolos inútiles, podemos determinar cuándo un lenguaje incontextual es finito o no.

Si dibujamos un gráfico con un vértice para cada variable de la gramática y una arista de  $A$  a  $B$  si hay una regla con forma  $A \rightarrow BC$  o  $A \rightarrow CB$  para cualquier  $C$ , cuando el gráfico resultante no tiene ciclos (es acíclico), el lenguaje generado por la gramática es finito (y si tiene ciclos, es un lenguaje infinito).

 El test de aciclicidad es un algoritmo clásico que se trata en la asignatura Estructura de la información.

Otra cuestión, quizá más importante, a la que también sabemos responder, es si una palabra dada pertenece a un lenguaje incontextual.

Una posible manera de determinarlo, partiendo de la gramática en forma normal de Greibach, sería generar de forma sistemática todas las derivaciones del mismo número de pasos que la palabra en cuestión y comprobar si alguna de las palabras generadas es la nuestra. Es un algoritmo muy deficiente, de tiempo exponencial respecto a la longitud de la palabra.

#### Recordad que...

... las reglas de una gramática en forma normal de Greibach añaden exactamente un símbolo terminal en un paso de derivación.

Existen varios algoritmos de tiempo cúbico (respecto a la longitud de la palabra) que deciden sobre la pertenencia de una palabra a un lenguaje incontextual. Uno de los más conocidos es el **algoritmo de Cocke-Kasami-Younger**.

Veremos este algoritmo en la versión que trabaja sobre una gramática  $G$  en forma normal de Chomsky y una palabra  $w$  (de longitud  $n$ ):

```

para  $i := 0$  hasta  $n - 1$  hacer
     $T_{i, i+1} := \emptyset$ ;
    para  $A \rightarrow a$  una producción de  $G$  hacer
        si  $a = w_{i+1}$  entonces  $T_{i, i+1} := T_{i, i+1} \cup \{A\}$  fsi
    fpara
fpara;
para  $m := 2$  hasta  $n$  hacer
    para  $i := 0$  hasta  $n - m$  hacer
         $T_{i, i+m} := \emptyset$ ;
        para  $j := i + 1$  hasta  $i + m - 1$  hacer
            para  $A \rightarrow BC$  una producción de  $G$  hacer
                si  $B \in T_{i, j} \wedge C \in T_{j, i+m}$  entonces  $T_{i, i+m} := T_{i, i+m} \cup \{A\}$  fsi
            fpara
        fpara
    fpara
fpara

```

Es un algoritmo en el que se aplica la técnica algorítmica conocida como *programación dinámica*, y que básicamente intenta ahorrar cálculos redundantes.

El algoritmo construye la tabla  $T$  en función de las reglas de la gramática  $G$  y la palabra  $w$ . La entrada  $i, j$  de  $T$  (que denotamos en el algoritmo  $T_{ij}$ ) contendrá al final el conjunto de variables de  $G$  que generan la subpalabra  $w_{ij}$  de  $w$ . Por tanto, si  $T_{0n}$  contiene  $S$  (el símbolo de inicio), podemos concluir que  $w$  es generada por  $G$ .

#### $w_{ij}$

Sea la palabra  $w = a_1 a_1 \dots a_n$ ; entonces, definimos la subpalabra  $w_{ij}$  como aquella que empieza con el símbolo  $i + 1$  y llega hasta el símbolo  $j$ ; es decir,  $w_{ij} = a_{i+1} \dots a_j$ .

Para saber si la palabra  $((()))$  pertenece al lenguaje de las cadenas no nulas de paréntesis balanceados, necesitamos tener su gramática en forma normal de Chomsky:

$$\begin{aligned} S &\rightarrow AB \mid AC \mid SS, \\ C &\rightarrow SB, \\ A &\rightarrow (, \\ B &\rightarrow ). \end{aligned}$$

La tabla  $T$  después de aplicar el algoritmo respecto a esta gramática y esta palabra es la siguiente:

$T$	1	2	3	4	5	6
0	$A$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$S$
1	—	$A$	$S$	$\emptyset$	$S$	$C$
2	—	—	$B$	$\emptyset$	$\emptyset$	$\emptyset$
$T$	1	2	3	4	5	6
3	—	—	—	$A$	$S$	$C$
4	—	—	—	—	$B$	$\emptyset$
5	—	—	—	—	—	$B$

La primera doble iteración inicializa la principal diagonal de la tabla  $T$  a partir de las dos últimas reglas (ponemos las variables que son parte izquierda de una regla que tiene como parte derecha la posición  $i$ -ésima de la palabra  $w$ ).

En la segunda iteración vamos llenando las diagonales que hay por encima de la principal a partir de los valores que tenemos en las diagonales anteriores. Al final llenamos la posición  $T[0, 6]$  (en general,  $T[0, n]$ ), que corresponde a la última diagonal.

Puesto que  $S$  pertenece a  $T_{0n}$ , podemos afirmar que la palabra dada pertenece al lenguaje generado por la gramática dada.

Para acabar, mencionamos alguna de las cuestiones sobre lenguajes incontextuales que no tienen una solución algorítmica.

Entre las preguntas sobre lenguajes incontextuales a las que ningún algoritmo puede responder (y que se denominan **problemas indecidibles**), tenemos las siguientes:

- Si dos gramáticas dadas son equivalentes (generan el mismo lenguaje).
- Si el complementario de un lenguaje incontextual dado es también incontextual.
- Si una gramática incontextual dada es ambigua.

## Resumen

Hemos empezado este módulo con la definición de **gramática generativa** (conjunto de reglas, cuya aplicación repetida nos genera un conjunto de palabras) y de **derivación** (cuál es el proceso que se debe seguir para generar una palabra), que nos ha llevado a la definición de **lenguaje generado por una gramática**. Para situarnos, hemos introducido la **clasificación de lenguajes de Chomsky** (respecto a las restricciones de las gramáticas que los generan) y nos hemos centrado en una de estas clases: los **lenguajes incontextuales** y las gramáticas que los generan, las **gramáticas incontextuales**.

En el segundo apartado hemos presentado los **árboles de derivación** para ayudarnos a decidir cuándo una gramática incontextual es ambigua.

El tercer apartado lo hemos dedicado básicamente a demostrar formalmente que una cierta gramática generaba realmente el lenguaje incontextual deseado. Mediante un ejemplo hemos mostrado cómo se debe realizar la verificación de gramáticas.

En los apartados cuarto y quinto hemos hablado de la **transformación de gramáticas incontextuales** en ciertos formatos con distintos objetivos: eliminación de reglas vacías y unitarias (**gramática limpia**), eliminación de variables y reglas inútiles (**gramática pelada**), y restricciones en los formatos de las reglas (**formas normales de Chomsky y Greibach**). En todos los casos hemos proporcionado métodos para realizar la transformación para asegurar que las gramáticas resultantes son equivalentes a las de partida.

Después de todo el trabajo sobre gramáticas incontextuales de los apartados anteriores, hemos definido los **autómatas con pila**, una máquina abstracta que permite reconocer lenguajes incontextuales. A continuación hemos mostrado la equivalencia de poder expresivo que hay entre los autómatas con pila y las gramáticas incontextuales.

Para acabar, en el séptimo y último apartado hemos dado un repaso a las **principales propiedades de los lenguajes incontextuales**: el lema del bombeo, que nos proporciona una forma de demostrar que un lenguaje no es incontextual, y el comportamiento de los lenguajes incontextuales respecto de las operaciones con lenguajes (unión, intersección, etc.). Y hemos diferenciado las preguntas sobre lenguajes incontextuales que tienen respuesta de las que no las tienen.



## Ejercicios de autoevaluación

1. Indicad gramáticas para los siguientes lenguajes incontextuales:

- a)  $\{a^i b^j c^j d^i \mid i, j \geq 1\}$ .
- b)  $\{a^i b^j \mid i > j \geq 0\}$ .
- c)  $\{w \in (a + b)^* \mid |w|_a = 2|w|_b\}$ .
- d)  $\{a^i b^j c^k \mid i \neq j \vee j \neq k\}$ .
- e)  $\{a^i b^j a^k \mid k = i + j\}$ .
- f)  $\{a^i b^j c^k \mid i = j \vee j = k\}$ .
- g)  $\{a^i b^j \mid i = j \vee j = 2i\}$ .
- h)  $\{a^i b^j c^k a^i \mid i \geq 1, j \geq k \geq 1\}$ .
- i)  $\{a^i b^j c^j d^i \mid i, j \geq 1\} \cup \{a^i b^j c^j d^i \mid i, j \geq 1\}$ .
- j)  $\{a^i b^j c^k d^l \mid i = k \vee j = l\}$ .
- k)  $\{a^{i+3} b^{2i+1} \mid i \geq 0\} \cup \{a^{2i+1} b^{3i} \mid i \geq 0\}$ .

2. Indicad cuáles de las siguientes gramáticas incontextuales son ambiguas:

- a)  $S \rightarrow aS \mid bT$   
 $T \rightarrow bS \mid aT \mid \lambda$ .
- b)  $S \rightarrow aSbSa \mid \lambda$ .
- c)  $S \rightarrow aSbSb \mid \lambda$ .

3. Encontrad una gramática incontextual pelada equivalente a la gramática siguiente:

$$\begin{aligned} S &\rightarrow AB \mid CA, \\ A &\rightarrow a, \\ B &\rightarrow BC \mid AB, \\ C &\rightarrow aB \mid b. \end{aligned}$$

4. Convertid a la forma normal de Chomsky la siguiente gramática incontextual:

$$\begin{aligned} S &\rightarrow AcB, \\ A &\rightarrow \lambda \mid aA, \\ B &\rightarrow \lambda \mid bB. \end{aligned}$$

5. Encontrad una gramática en forma normal de Greibach equivalente a la siguiente gramática incontextual:

$$\begin{aligned} S &\rightarrow 0 \mid AA, \\ A &\rightarrow 1 \mid SS. \end{aligned}$$

6. Construid autómatas con pila para los siguientes lenguajes incontextuales:

- a)  $\{a^i b^j \mid i > j \geq 0\}$ .
- b)  $\{w \in (a + b)^* \mid |w|_a \neq |w|_b\}$ .
- c)  $\{ba^{i_1} ba^{i_2} \dots ba^{i_j} \dots ba^{i_k} \mid k \geq 1, \exists j \text{ tal que } i_j = j\}$ .

7. Demostrad que el lenguaje  $L = \{a^i b^j c^j d^i \mid i, j \geq 1\}$  no es incontextual.

## Solucionario

### Ejercicios de autoevaluación

1.

a) Consiste en la concatenación de dos lenguajes (que son el mismo con símbolos terminales diferentes) muy conocidos:

$$\begin{aligned} S &\rightarrow AB, \\ A &\rightarrow aAb \mid ab, \\ B &\rightarrow cBd \mid cd. \end{aligned}$$

b) Debemos conseguir más  $a$  que  $b$ ; ponemos las mismas y dejamos poner todas las  $a$  de más que se desee, pero con la tercera regla aseguramos que como mínimo haya una más:

$$S \rightarrow aSb \mid aS \mid a.$$

c) Siempre el doble de  $a$  que de  $b$  (consultad el apartado g) de este ejercicio), pero en cualquier orden:

$$\begin{aligned} S &\rightarrow aCS \mid aBAS \mid bAAS \mid \lambda, \\ A &\rightarrow bAAA \mid a, \\ B &\rightarrow aCB \mid b, \\ C &\rightarrow aCC \mid aB. \end{aligned}$$

d) Claramente, el lenguaje se puede expresar como la unión de cuatro lenguajes:

$$\{a^i b^j c^k \mid i \neq j \vee j \neq k\} = \{a^i b^j c^k \mid i > j\} \cup \{a^i b^j c^k \mid i < j\} \cup \{a^i b^j c^k \mid j > k\} \cup \{a^i b^j c^k \mid j < k\},$$

y, por tanto, debemos dar una gramática para cada uno y relacionarlos como alternativa en la variable de inicio  $S$  ( $S_1$  sería la variable inicial del primer lenguaje de la unión;  $S_2$ , la del segundo;  $S_3$ , la del tercero, y  $S_4$ , la del cuarto):

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \mid S_3 \mid S_4, \\ S_1 &\rightarrow AC, \\ A &\rightarrow aAb \mid aA \mid a, \\ C &\rightarrow \lambda \mid cC. \\ S_2 &\rightarrow BC, \\ B &\rightarrow aBb \mid Bb \mid b. \\ S_3 &\rightarrow DE, \\ D &\rightarrow aD \mid \lambda \text{ para cuando } i = 0, \\ E &\rightarrow bEc \mid bE \mid b. \\ S_4 &\rightarrow DF, \\ F &\rightarrow bFc \mid Fc \mid c. \end{aligned}$$

Observad que aprovechamos la gramática del apartado b) de este mismo ejercicio.

e) Ponemos el mismo número de  $a$  delante y detrás, y después tantas  $b$  como  $a$  adicionales:

$$\begin{aligned} S &\rightarrow aSa \mid A, \\ A &\rightarrow \lambda \mid bAa. \end{aligned}$$

f) Se trata de la unión de dos lenguajes ya conocidos:

$$\begin{aligned} S &\rightarrow BC \mid AD, \\ A &\rightarrow aA \mid \lambda, \\ B &\rightarrow aBb \mid \lambda, \\ C &\rightarrow Cc \mid \lambda, \\ D &\rightarrow bDc \mid \lambda. \end{aligned}$$

g) También una unión de dos lenguajes, el primero conocido y el segundo deducible del primero:

$$\begin{aligned} S &\rightarrow A \mid B, \\ A &\rightarrow aAb \mid \lambda, \\ B &\rightarrow aBbb \mid \lambda. \end{aligned}$$

h) En primer lugar, generamos el mismo número de  $a$  delante y detrás de la palabra, y después controlamos más o igual número de  $b$  que  $c$ :

$$\begin{aligned} S &\rightarrow aSa \mid aAa, \\ A &\rightarrow bAc \mid bA \mid bc. \end{aligned}$$

i) Consiste en la unión de la gramática del apartado a) y una que hace crecer el mismo número de  $a$  que de  $d$ , y entre éstas, tantas  $b$  como  $c$ :

$$\begin{aligned} S &\rightarrow S_1 \mid S_2, \\ S_1 &\rightarrow AB, \\ A &\rightarrow aAb \mid ab, \\ B &\rightarrow cBd \mid cd, \\ S_2 &\rightarrow aS_2d \mid aCd, \\ C &\rightarrow bCc \mid bc. \end{aligned}$$

j) Debemos tratar por separado cuando  $i = k = 0$  dejando libre el número de  $b$  y  $d$  ( $BD$ ), y cuando  $j = l = 0$  dejando libre el número de  $a$  y  $c$  ( $AC$ ):

$$\begin{aligned} S &\rightarrow ED \mid AF \mid BD \mid AC, \\ E &\rightarrow aEc \mid aBc \mid \lambda, \\ F &\rightarrow bFd \mid bCd \mid \lambda, \\ A &\rightarrow aA \mid \lambda, \\ B &\rightarrow bB \mid \lambda, \\ C &\rightarrow cC \mid \lambda, \\ D &\rightarrow dD \mid \lambda. \end{aligned}$$

k) Es la unión de dos lenguajes:

$$\begin{aligned} S &\rightarrow S_1 \mid S_2, \\ S_1 &\rightarrow aaaAb, \\ A &\rightarrow aAbb \mid \lambda, \\ S_2 &\rightarrow aB, \\ B &\rightarrow aaBbbb \mid \lambda. \end{aligned}$$

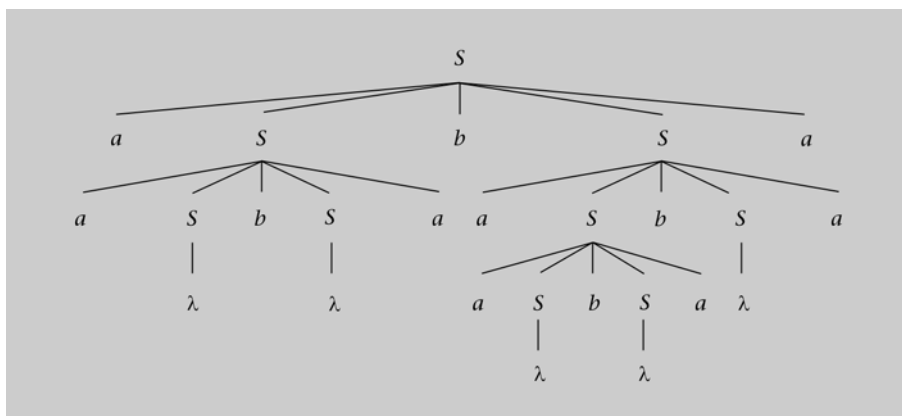
2. Recordad que para demostrar que una gramática es ambigua sólo es necesario encontrar una palabra para la que se puedan construir dos árboles de derivación, y para demostrar que no lo es, es necesario hacerlo en el caso general (para toda palabra).

a) No es ambigua. Desarrollamos el caso general para cada una de las variables ( $S$  con dos posibilidades y  $T$  con tres):



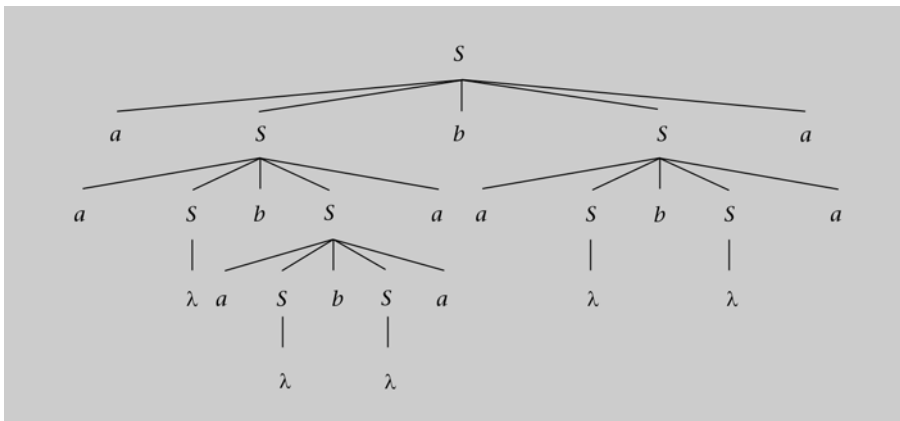
En todos los casos, observamos que, en un determinado momento, el hecho de aplicar sobre  $S$  o  $T$  una regla diferente hace que ya no pueda salir la misma palabra, ya sea porque hemos introducido una letra diferente o bien porque, como sucede en el último caso, nos lleva a palabras más cortas.

b) Sí que es ambigua. Por ejemplo, para  $m = aababaababaa$  podemos construir dos árboles:

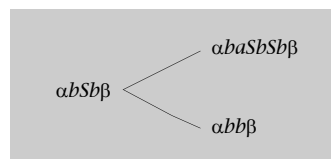
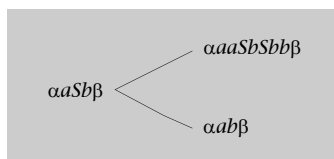


#### Observad que...

... se puede simplificar la palabra general, ya que la variable siempre ocupa la última posición.



c) No es ambigua. Desarrollamos el caso general para la variable única,  $S$ , teniendo en cuenta que hay dos posibles contextos para  $S$  ( $aSb$  y  $bSb$ ) a la vista de la primera regla:



#### Recordad que...

... cada alternativa representaba cada regla con la variable como parte izquierda.

En ambos casos observamos que introducimos diferencias irreconciliables entre las dos palabras generadas al aplicar en un determinado momento la primera o la segunda regla:  $aa$  frente a  $ab$ , y  $ba$  frente a  $bb$ , respectivamente.

3. La gramática incontextual de partida es la siguiente:

$$\begin{aligned} S &\rightarrow AB \mid CA, \\ A &\rightarrow a, \\ B &\rightarrow BC \mid AB, \\ C &\rightarrow aB \mid b. \end{aligned}$$

Para encontrar la gramática pelada damos los dos pasos siguientes:

- Aplicamos el método iterativo para encontrar el conjunto de variables útiles (coaccesibles):

$$\begin{aligned} &\{A, C\}, \\ &\{S, A, C\}. \end{aligned}$$

La variable  $B$  es inútil; a partir de ésta no se puede llegar a ninguna palabra.

Prescindiendo de  $B$  obtenemos la siguiente gramática equivalente:

$$\begin{aligned} S &\rightarrow CA, \\ A &\rightarrow a, \\ C &\rightarrow b. \end{aligned}$$

- El segundo método iterativo identifica las variables que quedan inútiles (accesibles):

$$\begin{aligned} &\{S\}, \\ &\{S, A, C\}. \end{aligned}$$

Dado que todas son accesibles, la gramática se puede considerar pelada:

$$\begin{aligned} S &\rightarrow CA, \\ A &\rightarrow a, \\ C &\rightarrow b. \end{aligned}$$

Después de pelar la gramática nos damos cuenta de que escondía un lenguaje muy simple, formado por una sola palabra,  $L = \{ba\}$ .

4. La gramática incontextual de partida es:

$$\begin{aligned} S &\rightarrow AcB, \\ A &\rightarrow \lambda \mid aA, \\ B &\rightarrow \lambda \mid bB. \end{aligned}$$

Si os fijáis, es como un procedimiento recursivo sin base. No acaba nunca.

Para encontrar su forma normal de Chomsky actuamos de la siguiente forma:

1) En primer lugar, debemos limpiar la gramática y, puesto que no hay reglas unitarias, sólo es necesario eliminar las producciones vacías:

a) Las variables anulables son  $A$  y  $B$  (pueden derivar la palabra vacía).

b) Si añadimos las reglas fruto de la ampliación, obtenemos la siguiente gramática:

$$\begin{aligned} S &\rightarrow AcB \mid cB \mid Ac \mid c, \\ A &\rightarrow \lambda \mid aA \mid a, \\ B &\rightarrow \lambda \mid bB \mid b. \end{aligned}$$

c) Ahora ya podemos sacar las reglas vacías (hemos incorporado directamente lo que no podíamos hacer sin éstas) y queda:

$$\begin{aligned} S &\rightarrow AcB \mid cB \mid Ac \mid c, \\ A &\rightarrow aA \mid a, \\ B &\rightarrow bB \mid b. \end{aligned}$$

2) Renombramos las variables  $A$  y  $B$  en  $D$  y  $E$  para poder introducir después las variables  $A$ ,  $B$  y  $C$  (no hay un conflicto de nombres) para sustituir a  $a$ ,  $b$  y  $c$ , respectivamente:

$$\begin{aligned} S &\rightarrow DCE \mid CE \mid DC \mid C, \\ D &\rightarrow AD \mid a, \\ E &\rightarrow BE \mid b, \\ A &\rightarrow a, \\ B &\rightarrow b, \\ C &\rightarrow c. \end{aligned}$$

#### Observad que...

... las  $a$ ,  $b$  y  $c$  solas de la última regla de  $S$ ,  $D$  y  $E$ , respectivamente, no se sustituyen por  $A$ ,  $B$  y  $C$  porque ya son de tipo 2 (leed la definición de *forma normal de Chomsky* en el subapartado 5.1.).

Todas las reglas están en forma normal de Chomsky, excepto la primera regla de  $S$ .

3) Introducimos una nueva variable  $F$  para dividir en dos esta primera regla y ya tenemos la gramática en forma normal de Chomsky:

$$\begin{aligned} S &\rightarrow DF \mid CE \mid DC \mid C, \\ F &\rightarrow CE, \\ D &\rightarrow AD \mid a, \\ E &\rightarrow BE \mid b, \\ A &\rightarrow a, \\ B &\rightarrow b, \\ C &\rightarrow c. \end{aligned}$$

Observad que es equivalente a la gramática de partida porque ésta no generaba la palabra vacía.

5. La gramática incontextual de partida es:

$$\begin{aligned} S &\rightarrow 0 \mid AA, \\ A &\rightarrow 1 \mid SS. \end{aligned}$$

Los pasos para obtener la forma normal de Greibach son los siguientes:

- Puesto que ya está en forma normal de Chomsky (dos reglas de tipo 1 y dos de tipo 2), sólo es necesario renombrar las variables:

$$\begin{aligned} A_1 &\rightarrow 0 \mid A_2A_2, \\ A_2 &\rightarrow 1 \mid A_1A_1. \end{aligned}$$

- Sólo hay una regla que no cumple  $j > i$ , la última. Trátemosla:

$$A_2 \rightarrow A_1A_1 \text{ se sustituye por } A_2 \rightarrow 0A_1 \text{ y } A_2 \rightarrow A_2A_2A_1,$$

pero en la segunda introducida nos encontramos con un caso  $i = j$ , en el que era preciso introducir una nueva variable  $B_2$  y las siguientes reglas:

$$\begin{aligned} B_2 &\rightarrow A_2A_1, \\ B_2 &\rightarrow A_2A_1B_2, \\ A_2 &\rightarrow 1B_2, \\ A_2 &\rightarrow 0A_1B_2. \end{aligned}$$

La gramática resultante de este paso será:

$$\begin{aligned} A_1 &\rightarrow 0 \mid A_2 A_2, \\ A_2 &\rightarrow 1 \mid 0 A_1 \mid 1 B_2 \mid 0 A_1 B_2, \\ B_2 &\rightarrow A_2 A_1 \mid A_2 A_1 B_2. \end{aligned}$$

- Sólo la segunda regla de  $A_1$  (de las variables  $A_i$ ) no está en forma normal de Greibach. Puesto que empieza con  $A_2$ , se debe sustituir por todas las posibles partes derechas de éstas (que son cuatro):

$$\begin{aligned} A_1 &\rightarrow 0 \mid 1 A_2 \mid 0 A_1 A_2 \mid 1 B_2 A_2 \mid 0 A_1 B_2 A_2, \\ A_2 &\rightarrow 1 \mid 0 A_1 \mid 1 B_2 \mid 0 A_1 B_2, \\ B_2 &\rightarrow A_2 A_1 \mid A_2 A_1 B_2. \end{aligned}$$

- Finalmente, debemos poner en forma normal las reglas de  $B_2$ , sustituyendo la primera variable de sus partes derechas ( $A_2$  en los dos casos) por todas las partes derechas en que  $A_2$  es parte izquierda:

$$\begin{aligned} A_1 &\rightarrow 0 \mid 1 A_2 \mid 0 A_1 A_2 \mid 1 B_2 A_2 \mid 0 A_1 B_2 A_2, \\ A_2 &\rightarrow 1 \mid 0 A_1 \mid 1 B_2 \mid 0 A_1 B_2, \\ B_2 &\rightarrow 1 A_1 \mid 0 A_1 A_1 \mid 1 B_2 A_1 \mid 0 A_1 B_2 A_1 \mid 1 A_1 B_2 \mid 0 A_1 A_1 B_2 \mid 1 B_2 A_1 B_2 \mid 0 A_1 B_2 A_1 B_2. \end{aligned}$$

6.

a) La idea es que debemos consumir las  $a$  y apilarlas y, cuando lleguen las  $b$ , desapilar una  $a$  por cada  $b$  consumida. Cuando acabemos de consumir la palabra, debe quedar como mínimo una  $a$  en la pila.

**Observad que no debemos aceptar la palabra vacía.**

$$\begin{aligned} (q_0, a, z_0) &\vdash (q_0, a z_0), \\ (q_0, \lambda, a) &\vdash (q_f, \lambda), \\ (q_0, a, a) &\vdash (q_0, a a), \\ (q_0, b, a) &\vdash (q_1, \lambda), \\ (q_1, b, a) &\vdash (q_1, \lambda), \\ (q_1, \lambda, a) &\vdash (q_f, \lambda), \\ (q_f, \lambda, a) &\vdash (q_f, \lambda), \\ (q_f, \lambda, z_0) &\vdash (q_f, \lambda). \end{aligned}$$

Para que no haya ninguna duda, hemos dado un autómata que acepta por estado final y por pila vacía (en el estado final, desapilamos las  $a$  que quedan).

b) Puesto que debemos comprobar que el número de  $a$  es diferente del número de  $b$  pero  $a$  y  $b$  pueden estar mezcladas, guardaremos siempre en la pila los elementos de más hasta aquel momento (si ganan las  $a$ , habrá  $a$ ; si hay más  $b$ , habrá  $b$ ; si hay el mismo número de  $a$  y  $b$ , la pila estará vacía). Cuando acabemos de consumir la palabra, la pila no puede estar vacía (tiene que haber como mínimo una  $a$  o una  $b$ ):

$$\begin{aligned} (q_0, a, z_0) &\vdash (q_0, a z_0), \\ (q_0, b, z_0) &\vdash (q_0, b z_0), \\ (q_0, a, a) &\vdash (q_0, a a), \\ (q_0, a, b) &\vdash (q_0, \lambda), \\ (q_0, b, b) &\vdash (q_0, b b), \\ (q_0, b, a) &\vdash (q_0, \lambda), \\ (q_0, \lambda, a) &\vdash (q_f, \lambda), \\ (q_0, \lambda, b) &\vdash (q_f, \lambda), \\ (q_f, \lambda, a) &\vdash (q_f, \lambda), \\ (q_f, \lambda, b) &\vdash (q_f, \lambda), \\ (q_f, \lambda, z_0) &\vdash (q_f, \lambda). \end{aligned}$$

También acepta por estado final y pila vacía: después de consumir el último símbolo y comprobar que hay una  $a$  o una  $b$  en la pila, pasamos al estado final y vaciamos la pila.

c) Debemos apilar las  $b$ , saltar las secuencias de  $a$  y, en un determinado momento (de forma no determinista), tenemos que comprobar que el número de  $b$  apiladas se corresponde con el número de  $a$  de la secuencia correspondiente:

$$\begin{aligned} (q_0, b, z_0) &\vdash (q_0, b z_0), \\ (q_0, b, z_0) &\vdash (q_1, b z_0), \\ (q_1, a, b) &\vdash (q_1, \lambda), \\ (q_1, b, z_0) &\vdash (q_f, z_0), \\ (q_1, \lambda, z_0) &\vdash (q_f, \lambda), \\ (q_0, a, b) &\vdash (q_0, b), \\ (q_0, b, b) &\vdash (q_0, b b), \end{aligned}$$

**Atención**

Es preciso tratar el primer bloque aparte (corresponde a las cinco primeras transiciones).

$$\begin{array}{ll}
(q_0, b, b) & \vdash (q_1, bb), \\
(q_1, a, b) & \vdash (q_1, \lambda), \\
(q_1, b, b) & \vdash (q_f, z_0), \\
(q_1, \lambda, z_0) & \vdash (q_f, \lambda). \\
(q_f, a, z_0) & \vdash (q_f, z_0), \\
(q_f, b, z_0) & \vdash (q_f, z_0), \\
(q_f, \lambda, z_0) & \vdash (q_f, \lambda).
\end{array}$$

En **negrita** hemos indicado los dos pares de transiciones que representan la elección no determinista entre saltarse el bloque (primeras transiciones, en las que quedamos con el mismo estado) o comprobar que es el buen bloque (segundas transiciones, en las que pasamos al estado de verificación de que el número de  $a$  hasta la próxima  $b$  es igual al número de  $b$  en la pila).

Las cinco primeras transiciones corresponden al tratamiento especial del primer bloque y podemos hacer dos cosas: o bien apilar la  $b$  y saltar las  $a$  o bien comprobar que el primer bloque es  $ba$  (y aceptar).

Las siguientes seis transiciones saltan las  $a$  y apilan las  $b$ , o bien pasan al estado de comprobación ( $q_1$ ) apilando la  $b$ , y desapilan una  $b$  por cada  $a$  (tercera transición del grupo anterior) y pasan al estado final, si al encontrar la próxima  $b$  hemos llegado al fondo de la pila.

Las tres últimas transiciones, ya en el estado final, consumen lo que queda de la palabra y al final vacían la pila.

7. Para el lema del bombeo podemos tomar la palabra del lenguaje:

$$m = a^N b^{N+1} c^N d^{N+1},$$

y, si consideramos las factorizaciones  $m = uvwxy$  tales que  $vx$  no vacía y la longitud de  $vw$  menor o igual que  $N$ , podemos considerar siete casos:

- a)  $v$  y  $x$  sólo tienen  $a$ : entonces  $uv^iwx^iy$  para  $i = 0$  ( $uvw$ ) como mínimo tiene una  $a$  menos que  $c$  (no afectadas por el bombeo) y la palabra no pertenece al lenguaje.
- b)  $v$  y  $x$  sólo tienen  $b$ : entonces  $uv^iwx^iy$  para  $i = 0$  ( $uvw$ ) como mínimo tiene una  $b$  menos que  $d$  (no afectadas por el bombeo) y la palabra no pertenece al lenguaje.
- c)  $v$  y  $x$  sólo tienen  $c$ : entonces  $uv^iwx^iy$  para  $i = 0$  ( $uvw$ ) como mínimo tiene una  $c$  menos que  $a$  (no afectadas por el bombeo) y la palabra no pertenece al lenguaje.
- d)  $v$  y  $x$  sólo tienen  $d$ : entonces  $uv^iwx^iy$  para  $i = 0$  ( $uvw$ ) como mínimo tiene una  $d$  menos que  $b$  (no afectadas por el bombeo) y la palabra no pertenece al lenguaje.
- e)  $v$  y  $x$  tienen  $a$  y  $b$ : entonces  $uv^iwx^iy$  para  $i = 0$  ( $uvw$ ) como mínimo tiene una  $a$  o  $b$  menos que  $c$  o  $d$  (no afectadas por el bombeo), respectivamente, y la palabra no pertenece al lenguaje.
- f)  $v$  y  $x$  tienen  $b$  y  $c$ : entonces  $uv^iwx^iy$  para  $i = 0$  ( $uvw$ ) como mínimo tiene una  $b$  o  $c$  menos que  $d$  o  $a$  (no afectadas por el bombeo), respectivamente, y la palabra no pertenece al lenguaje.
- g)  $v$  y  $x$  tienen  $c$  y  $d$ : entonces  $uv^iwx^iy$  para  $i = 0$  ( $uvw$ ) como mínimo tiene una  $c$  o  $d$  menos que  $a$  o  $b$  (no afectadas por el bombeo), respectivamente, y la palabra no pertenece al lenguaje.

En todos los casos posibles se puede apreciar que la palabra no pertenece al lenguaje y, para el lema del bombeo, podemos concluir que el lenguaje no es incontextual.

También se puede llegar a esta conclusión si intentamos pensar en el autómata con pila que tendría si fuera incontextual: apilaríamos las  $a$  para poder comprobar después que hay tantas como  $c$ ; después apilaríamos las  $b$  para comprobar que hay tantas como  $d$ , pero al llegar a las  $c$  no podríamos hacer la comprobación pertinente porque las  $a$  están debajo de las  $b$ , y a éstas no las podemos sacar de la pila porque entonces no podemos comprobar que hay tantas como  $d$ . Es imposible construir un autómata con pila para este lenguaje.

## Glosario

### alfabeto

Conjunto finito y no vacío de símbolos.

### algoritmo

Conjunto explícito de reglas para resolver un problema en un número finito de pasos.

**autómata con pila**

Máquina abstracta con memoria infinita (con estructura de pila) que permite reconocer lenguajes incontextuales.

**autómata finito**

Máquina abstracta con memoria finita que permite reconocer lenguajes regulares.

**depuración de gramáticas**

Proceso de eliminación de elementos inútiles de una gramática.

**derivación**

Proceso de aplicación de las reglas de una gramática.

**forma normal**

Restricciones especiales en las reglas de una gramática.

**gramática generativa**

Conjunto finito de reglas de producción cuya aplicación repetida nos proporciona un conjunto de palabras de un lenguaje.

**inversión**

Operación consistente en obtener una palabra como resultado de invertir el orden de los símbolos de otra palabra.

**lenguaje formal**

Conjunto de palabras.

**palabra**

Secuencia de símbolos de un alfabeto.

**problemas indecidibles**

Problemas sin solución algorítmica.

**regla de producción**

Pares ordenados que se utilizan para derivar nuevas palabras a partir de palabras ya dadas por medio de sustituciones.

**variable**

Símbolo no terminal de una gramática.

## Bibliografía

**Hopcroft, J.E.; Ullman, J.D.** (1979). *Introduction to Automata Theory, Languages and Computation*. Reading (Mass): Addison-Wesley.

Hay una traducción al castellano: *Introducción a la teoría de autómatas, lenguajes y computación*. 2.<sup>a</sup> edición (2002). México: CECSA.

**Isasi, P.; Martínez, P.; Borrajo, D.** (1997). *Lenguajes, gramáticas y autómatas*. Madrid: Addison-Wesley Iberoamericana.