Teoría de la Computación

UCM, semestre 2020-2

Pablo Sáez

pablosaezphd@gmail.com

La pregunta del millón, para comenzar

Qué viene siendo la teoría de la computación? En qué consiste?

La teoría de la computación

En realidad hay una serie de fundamentos teóricos para la informática, muchos de ellos basados en la lógica matemática. Se pueden considerar por ejemplo las bases de datos desde un punto de vista teórico, de la lógica, se estudia también la teoría de la computabilidad (por ejemplo qué viene siendo una función computable), se consideran las especificaciones formales del software, se realizan demostraciones matemáticas de la correctitud de un programa etc.

Es decir que el área es amplia.

En este ramo nos concentraremos en lo que se denomina "teoría de autómatas y lenguajes formales", y en la teoría de la computabilidad.

Problema preliminar

Antes de entrar en materia veamos un ejemplo concreto, para partir.

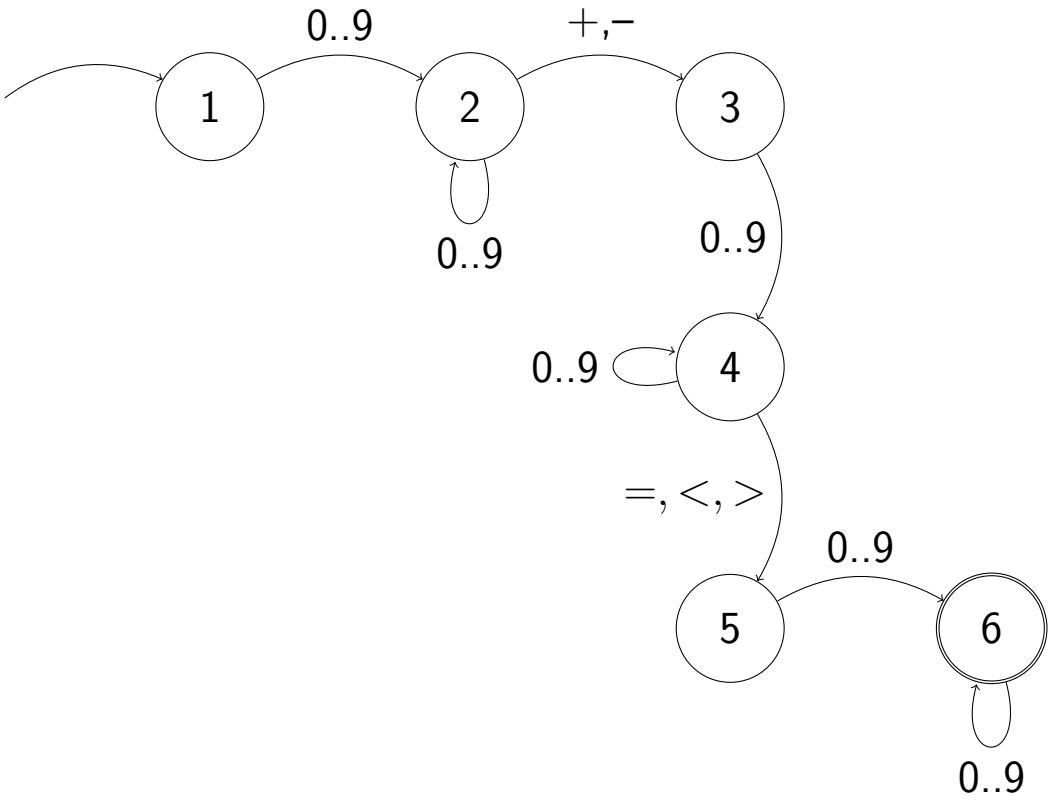
Cómo podríamos escribir un programa que reconozca expresiones matemáticas tales como las siguientes:

- $2 + 2 = 4$
- $20 - 40 > 0$
- $1 + 1 = 11$

es decir expresiones que incluyan a un entero, una suma o una resta, un entero, un comparador, y un entero, en ese orden.

Nótese que la pregunta tiene que ver con decir si la expresión está sintácticamente correcta, no si es verdadera o falsa.

Solución: un autómata finito



Qué es la teoría de autómatas y lenguajes formales?

- Veíamos la clase pasada un problema de reconocimiento de expresiones matemáticas. Este problema puede ser resuelto por un autómata finito.
- Veremos a lo largo del curso los autómatas finitos, los autómatas con pila, las máquinas de Turing y otros.
- El término "autómata" se refiere a mecanismos de cómputo que procesan lenguajes formales.
- Entonces: qué es un lenguaje formal?

Los lenguajes formales: definiciones, ejemplos

- Un lenguaje es un conjunto de strings de letras. Las letras pertenecen a un conjunto finito llamado *alfabeto*.
- Ejemplo 1: los enteros binarios de 8 bits forman un lenguaje (que tiene 2^8 strings). El alfabeto es $\{0, 1\}$. Los strings del lenguaje son: `01100111`, `11111011` etc.
- Ejemplo 2: Las fórmulas matemáticas forman un lenguaje. Ejemplos de fórmulas: `x+2=3*y-1`,`f(z)>4`. El alfabeto es $\{x, 0, 1, 2, y, f, (,), 4, ...\}$ (finito). El lenguaje es en principio infinito (hay infinitas fórmulas que se pueden escribir).
- Ejemplo 3: El lenguaje de programación C. El alfabeto es el conjunto de caracteres ASCII.

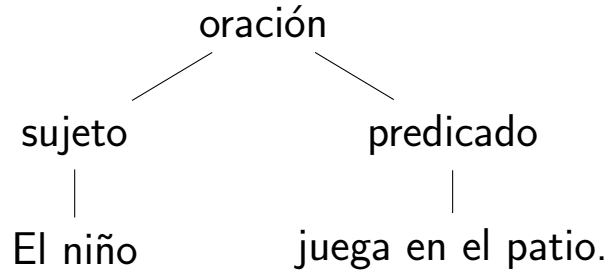
Los lenguajes formales (cont.)

- Un lenguaje es *formal* en la medida en que puede ser descrito mediante reglas de sintáxis formales, rigurosas. Por ejemplo los lenguajes anteriores son formales.
- El castellano no es un lenguaje formal.
- Pero lo usaremos como ejemplo didáctico para entender la problemática.
- En la asignatura de lenguaje en la escuela nos enseñaron la gramática de nuestro idioma.
- Tomemos como ejemplo la oración *El niño juega en el patio.*

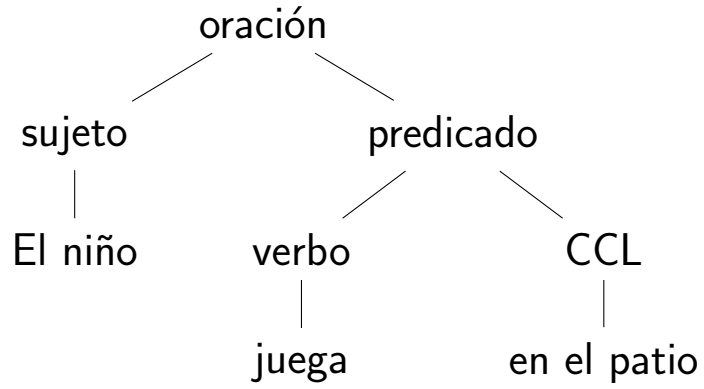
Recordemos la materia de lenguaje

Sabemos que la oración tiene sujeto y predicado. En el ejemplo el sujeto es "El niño" y el predicado es "juega en el patio".

Esquemáticamente:

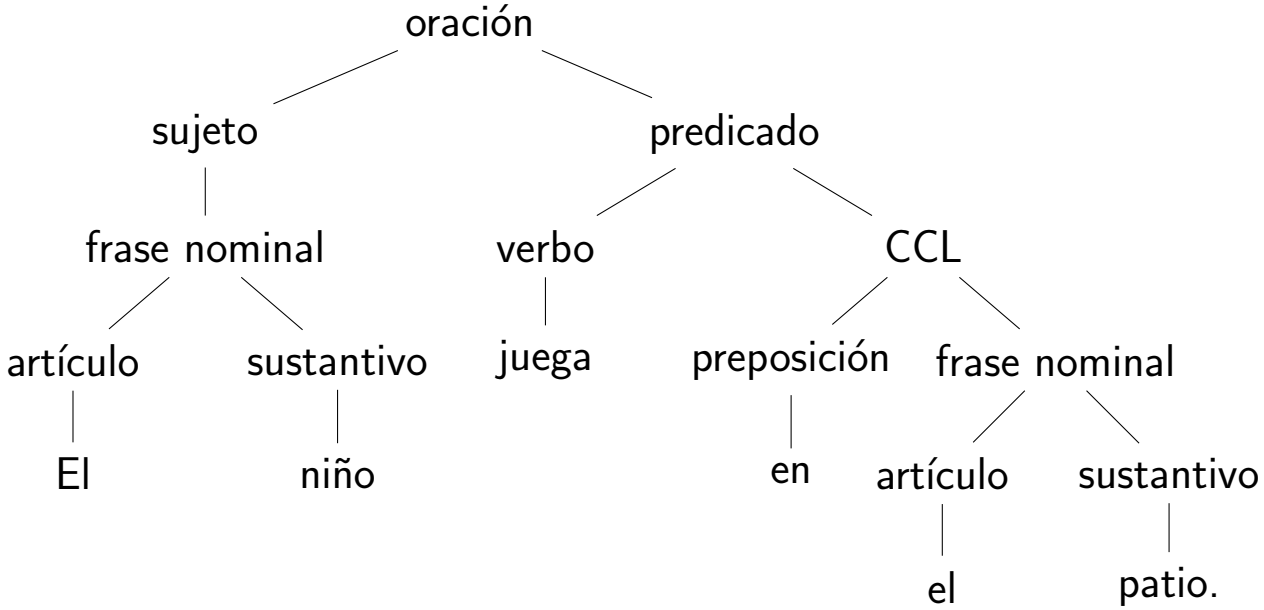


A su vez el predicado (en este ejemplo) consta de: verbo y complemento circunstancial de lugar (CCL). Es decir:



El ejemplo completo.

Yéndonos al final del asunto, la estructura gramatical de la frase anterior es:



La gramática.

Podemos pensar en un conjunto de *reglas gramáticales* para el ejemplo propuesto. Concretamente:

<oración> $\rightarrow$ <sujeto> <predicado>

<sujeto> $\rightarrow$ <frase nominal>

<frase nominal> $\rightarrow$ <artículo> <sustantivo>

<artículo> $\rightarrow$ el

<artículo> $\rightarrow$ un

<sustantivo> $\rightarrow$ niño

<sustantivo> $\rightarrow$ papá

<sustantivo> $\rightarrow$ patio

<sustantivo> $\rightarrow$ living

<predicado> $\rightarrow$ <verbo> <CCL>

<verbo> $\rightarrow$ juega

<verbo> $\rightarrow$ lee

<CCL> $\rightarrow$ <preposición> <frase nominal>

<preposición> $\rightarrow$ en

(distinguimos los símbolos no-terminales de los terminales poniendo a los primeros entre < >).

Aplicación de la gramática anterior.

La gramática anterior permite generar algunas oraciones del castellano. Aparte de "el niño juega en el patio" genera "el papá lee en el living", "un niño lee en un living", "el living juega en el patio", etc.

En realidad estas reglas permiten generar *algunas* oraciones del castellano. El idioma es más complejo que esto y no puede ser descrito completamente por reglas de este tipo, por ende nuestro idioma no es un lenguaje formal; pero el subconjunto del idioma que puede ser generado por reglas de este tipo sí se puede considerar un lenguaje formal. Por ejemplo el lenguaje de la página anterior es un lenguaje formal.

Pregunta: cómo podríamos escribir una gramática (formal, es decir del tipo de la página anterior) para el lenguaje de la clase pasada, el de las expresiones aritméticas?

Gramática: definición formal.

Formalmente, una *gramática libre del contexto* es una 4-tupla $G = (V, T, S, R)$ en donde V es un conjunto finito de símbolos llamados *no-terminales*, T es un conjunto finito de símbolos llamados *terminales*, S es un elemento de V, es decir $S \in V$, y R es un conjunto de reglas de la forma $a \rightarrow b_1 b_2 ... b_n$ en donde $a \in V$ y $b_i \in V \cup T$ para $i = 1...n$, es decir que el símbolo de la izquierda es un no-terminal y los símbolos de la derecha son terminales o no-terminales.

El *lenguaje generado por* $G = (V, T, S, R)$, denotado por $L(G)$, es el conjunto de strings que se obtienen partiendo de S, reescribiendo sucesivamente los símbolos no-terminales aplicando las reglas de R, en todas las combinaciones posibles, hasta que el string contenga solamente símbolos terminales.

Nótese que (a) el lenguaje generado por una gramática puede contener una cantidad infinita de strings, y (b) el *alfabeto* del lenguaje es precisamente T.

Ejemplo: $G = (V, T, S, R)$, con $V = \{s\}$, $T = \{a, b\}$, $S = s$, $R = \{s \rightarrow a, s \rightarrow sb\}$.

El lenguaje generado por G es $L(G) = \{a, ab, abb, abbb, ...\}$, como se puede comprobar. Es un lenguaje infinito.

Lenguajes: definiciones.

Hemos visto que un lenguaje es un conjunto de strings de letras de un alfabeto A. Vimos los lenguajes generados por gramáticas libres del contexto. No todos los lenguajes pueden ser generados por gramáticas libres del contexto.

En general, dados dos strings s_1 y s_2 con letras de un alfabeto A se define la operación $s_1 \cdot s_2$ como la *concatenación* de s_1 y s_2. Por ejemplo si $s_1 = abc$ y $s_2 = cd$ entonces $s_1 \cdot s_2 = abccd$. Se define ϵ como el string *vacío*. Es decir $s \cdot \epsilon = \epsilon \cdot s = s$.

Nótese que la concatenación es una ley de composición interna que es asociativa y posee un elemento neutro, es decir que viene siendo un *monoide* (como estructura algebraica, pero no ahondaremos en esto). Nótese además que ϵ *no es* un elemento del alfabeto A.

Se definen las *potencias* de un string del modo siguiente: $s^0 = \epsilon, s^1 = s, s^2 = s \cdot s, s^3 = s \cdot s \cdot s$ etc.

Se define la *concatenación* de dos lenguajes L_1 y L_2 como $L_1 \cdot L_2 = \{s_1 \cdot s_2, s_1 \in L_1 \wedge s_2 \in L_2\}$. Se definen las *potencias* de los lenguajes como $L^2 = L \cdot L, L^3 = L \cdot L \cdot L$ etc. Se define la *estrella de Kleene* como $L^* = \bigcup_{n \in \mathbb{N}} L^n$.

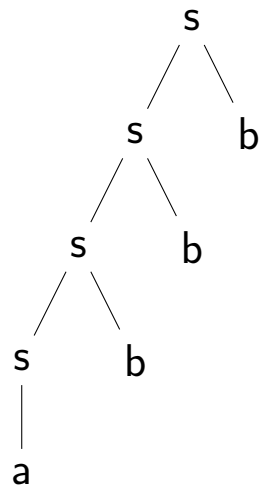
Nótese que para cualquier lenguaje L, $\epsilon \in L^0$, por ende $\epsilon \in L^*$.

Ejemplo: el lenguaje de la trasparencia anterior es $\{ab^n, n \in \mathbb{N}\}$.

Arbol de derivación

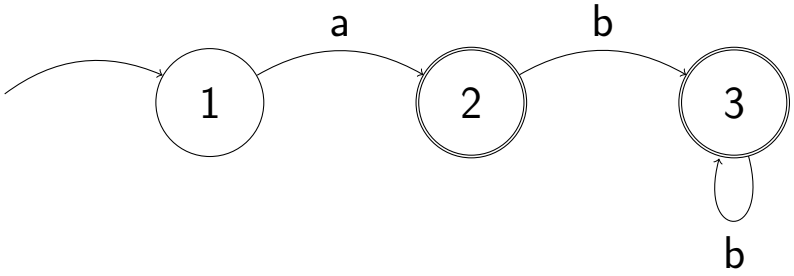
Dada una gramática $G = (V, T, S, R)$ y un string $s_1 \in L(G)$, se define el *árbol de derivación* de s_1 como el árbol que tiene a S en la raíz, de modo que los hijos de un nodo n se obtienen aplicando una regla $n \to h_1 h_2 ... h_n \in R$, es decir que los hijos de n son $h_1, ..., h_n$, y de modo que las *hojas* del árbol correspondan a las letras de s_1, en orden.

Veíamos en una transparencia anterior el árbol de derivación para la frase "El niño juega en el patio.". O sino para el ejemplo de la trasparencia anterior, el árbol de derivación de $abbb$ es:



Autómata finito: definición

Vimos en la primera trasparencia un ejemplo de autómata finito que reconoce un lenguaje. Un ejemplo más sencillo, para el lenguaje $L = \{ab^n, n \in \mathbb{N}\}$ es:



Formalmente, un autómata finito es una quíntupla $(S, \Sigma, \delta, \iota, F)$ en donde

- S es un conjunto finito de estados. En el ejemplo: $S = \{1, 2, 3\}$.
- Σ es un conjunto finito, el alfabeto del autómata. En el ejemplo: $\Sigma = \{a, b\}$.
- $\iota \in S$ es el estado inicial del autómata. En el ejemplo: $\iota = 1$.
- $F \subseteq S$ es el conjunto de estados finales, o estados de aceptación, del autómata. En el ejemplo: $F = \{2, 3\}$.
- $\delta : S \times \Sigma \to S$ es una función (parcial) llamada *función de transición*. En el ejemplo: $\delta(1, a) = 2, \delta(2, b) = 3, \delta(3, b) = 3$. El significado de $\delta(q, x) = p$ es "si el autómata está en el estado q y lee la letra x entonces pasa al estado p".

Lenguajes regulares

Decimos que un autómata $(S, \Sigma, \delta, \iota, F)$ acepta un string s (con letras de Σ) si, partiendo del estado ι y leyendo el string de s letra por letra utilizando transiciones de δ, llega a un estado de aceptación (un estado de F) habiendo leído el string s completo.

El lenguaje *reconocido* por un autómata A, $L(A)$, es el conjunto de los strings aceptados por A. Un lenguaje es *regular* (por definición) si y sólo si es aceptado por algún autómata finito.

Teorema: el conjunto de los lenguajes regulares es un subconjunto (propio) del conjunto de los lenguajes generados por gramáticas libres del contexto.

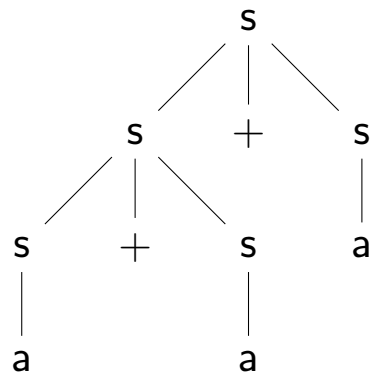
Gramáticas ambiguas

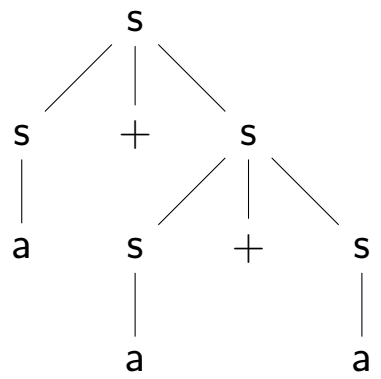
Decimos que una gramática G es *ambigua* si existe una palabra en el lenguaje generado por la gramática, $L(G)$ con dos árboles de derivación distintos.

Por ejemplo la siguiente gramática es ambigua

$$s \to s + s$$
$$s \to a$$

dado que el string $a + a + a$ puede ser generado de dos formas distintas:





Problemas con las gramáticas ambiguas

Desde un punto de vista práctico hay algunos problemas que pueden surgir cuando una gramática es ambigua. Por ejemplo la siguiente gramática también es ambigua:

$$
\begin{aligned}
s &\to s + s \\
s &\to s \times s \\
s &\to 1 \\
s &\to 2
\end{aligned}
$$

y el string $1 + 2 \times 1$ dependiendo de las reglas que se usen se puede interpretar como $(1 + 2) \times 2$ (o sea 6) o bien como $1 + (2 \times 2)$ (o sea 5).

Una gramática no ambigua para este lenguaje sería:

$$
\begin{aligned}
s &\to t \\
s &\to s + t \\
t &\to u \\
t &\to t \times u \\
u &\to 1 \\
u &\to 2
\end{aligned}
$$

Autómatas finitos no deterministas

Consideremos la gramática

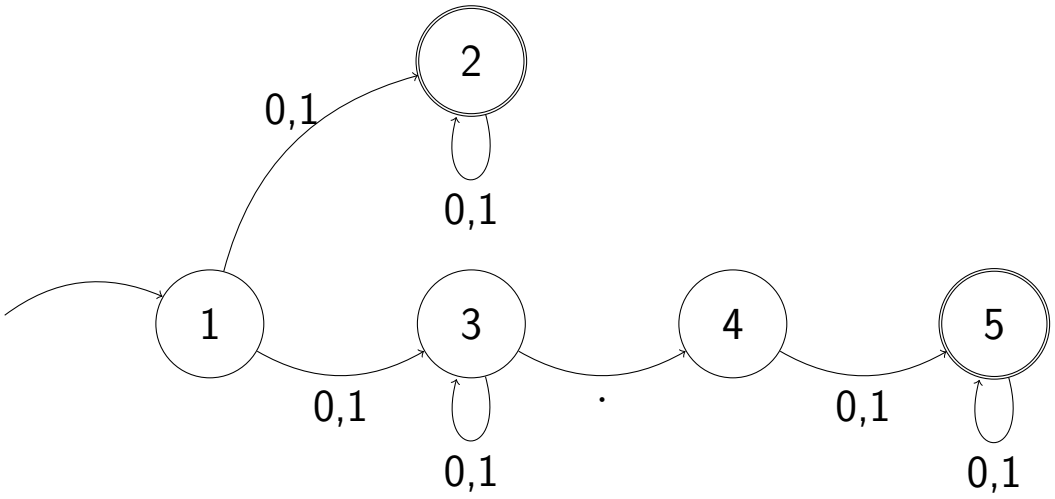
<numero>→<entero>|<decimal>

<entero>→<entero> <digito>|<digito>

<digito>→1|0

<decimal>→<entero> . <entero>

Uno podría pensar a partir de esa gramática en el siguiente autómata



Este autómata desde el estado 1 puede ir a dos estados distintos con el mismo input: al estado 2 si es un entero y al estado 3 si es un decimal. Esto hace que en el estado 1 haya un no determinismo, por lo tanto el autómata es un autómata finito no determinista (AFND).

AFND: definiciones

Formalmente, un autómata finito es una quíntupla $(S, \Sigma, \delta, \iota, F)$ en donde

- S es un conjunto finito de estados
- Σ es un conjunto finito, el alfabeto del autómata.
- $\iota \in S$ es el estado inicial del autómata.
- $F \subseteq S$ es el conjunto de estados finales, o estados de aceptación, del autómata.
- $\delta \subseteq S \times \Sigma \times S$ es una relación de transición. El significado de $\delta(q, x, p)$ es "si el autómata está en el estado q y lee la letra x entonces *puede* pasar al estado p".

Es decir que la diferencia entre los AFD y los AFND es que en vez de una función de transición hay una relación de transición.

Teorema: Un lenguaje es reconocido por un AFD si y sólo si es reconocido por un AFND.

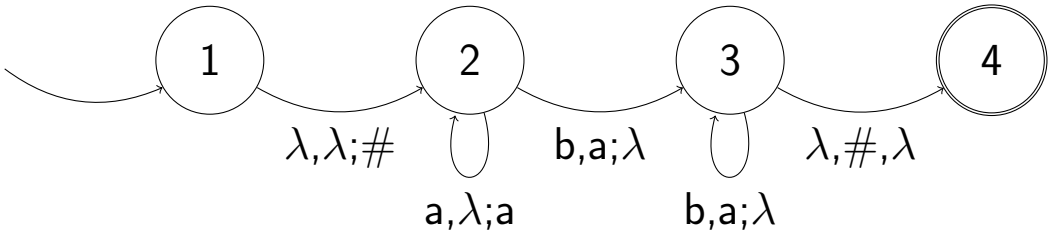
Es decir que el no-determinismo en este caso no aporta un mayor "poder de cálculo". Podemos decir entonces que un lenguaje es regular si es aceptado por un AFD o por un AFND (cualquiera de los dos).

Autómatas con pila

Existen lenguajes que no pueden ser reconocidos por autómatas finitos, por ejemplo el siguiente: $\{a^n b^n, n \in \mathbb{N}\}$.

En efecto, el autómata tendría que ir leyendo las letras a y *recordando* cuantas lleva leídas para después leer las letras b, lo cual no es posible para un autómata finito, dado que no tiene memoria.

Este problema puede ser resuelto con un autómata con pila:



En cada transición el primer símbolo es el que se lee desde el input, el segundo símbolo es que se desempila, y el tercer símbolo es el que se empila. El símbolo λ significa no hacer nada.

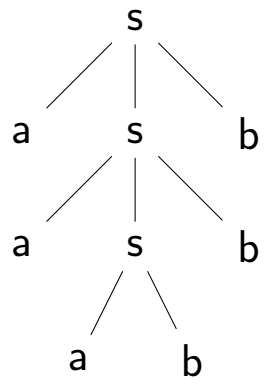
Pregunta: cómo sería una gramática para este lenguaje?

Solución del problema anterior.

Una gramática para el lenguaje $\{a^n b^n, n \in \mathbb{N}\}$ puede ser por ejemplo la siguiente:

$$s \to ab$$
$$s \to asb$$

Con esa gramática un árbol de derivación para el string $aaabbb$ sería:



Ejecución del autómata de pila del ejemplo

Consideremos el procesamiento del string $aaabbb$. Al comienzo el autómata está posicionado en la primera a, lo que puede ser representado así: $\underline{a}aabbb$, y la pila está vacía.

Estamos en el estado 1 y la primera transición es: $\lambda, \lambda; \#$. Es decir: no leer nada desde el input, no leer nada desde la pila, y empilar $\#$. Por lo tanto pasamos al estado 2, el input sigue igual, o sea $\underline{a}aabbb$, y en la pila tenemos: $\#$.

En el estado 2 la transición es $a, \lambda; a$, hacia el mismo estado 2. Es decir: se lee una a, no se desempila nada, y se empila una a. Por lo tanto el input queda en $a\underline{a}abbb$ y la pila queda en: $\#a$.

Luego la transición es la misma, es decir se efectúan las mismas operaciones, y quedamos con el input en $aa\underline{a}bbb$, y con la pila en $\#aa$.

Luego de nuevo lo mismo, quedamos en el estado 2 con el input en $aaa\underline{b}bb$ y la pila en $\#aaa$.

Ejecución del autómata (cont.)

Ahora leemos una b desde el input, por lo tanto la transición es $b, a; \lambda$ hacia el estado 3. Es decir: leer la b, desempilar una a, y no empilar nada. Quedamos entonces con $aaab\underline{b}b$ en el input, y con $\#aa$ en la pila.

Estamos ahora en el estado 3 y la transición es $b, a; \lambda$ hacia el mismo estado 3. El input queda en $aaabb\underline{b}$ y la pila queda en $\#a$.

Después de nuevo lo mismo, y quedamos en $aaabbb\_$ en el input (es decir que ya se leyó todo), y con $\#$ en la pila.

Finalmente pasamos del estado 3 al estado 4 (de aceptación) con la transición $\lambda, \#, \lambda$, es decir no leer nada desde el input, desempilar el $\#$ y no empilar nada.

Entonces:

1. Se ejecutaron transiciones del autómata de modo que
2. se leyó todo el input,
3. se llegó a un estado de aceptación, y
4. la pila quedó vacía.

Como se cumplen las condiciones 1, 2, 3, y 4 el autómata de pila *acepta* el input.

Autómatas con pila: definición formal

Formalmente, un autómata con pila es una séxtupla de la forma $(S, \Sigma, \Gamma, T, \iota, F)$, donde

- S es un conjunto finito de estados. En el ejemplo $S = \{1, 2, 3, 4\}$.
- Σ es un conjunto finito, el alfabeto de la máquina. En el ejemplo $\Sigma = \{a, b\}$.
- Γ es un conjunto finito, el alfabeto de la pila. En el ejemplo $\Gamma = \{\#, a\}$.
- $T \subseteq S \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times S \times (\Gamma \cup \{\lambda\})$ es un conjunto finito de transiciones. La transición $(p, x, s; q, y) \in T$ se interpreta como sigue: "si el autómata está en el estado p, si el símbolo de entrada es x y si el símbolo que está en el tope de la pila es s, entonces lee x, desempila s, pasa al estado q y empila el símbolo y". Nótese que tanto x como s e y pueden ser λ, lo cual significa no leer, no empilar o no desempilar nada. En el ejemplo $T = \{(1, \lambda, \lambda; 2, \#), (2, a, \lambda; 2, a), (2, b, a; 3, \lambda), (3, b, a; 3, \lambda), (3, \lambda, \#, 4, \lambda)\}$
- ι es un elemento de S, es decir $\iota \in S$, el estado inicial. En el ejemplo $\iota = 1$.
- F es un subconjunto de S, es decir $F \subseteq S$, el conjunto de estados de aceptación. En el ejemplo $F = \{4\}$.

Jerarquía de Chomsky