Lenguajes Formales, Autómatas y Computabilidad

BAS

Resumen de la materia Lenguajes Formales, Autómatas y Computabilidad dictada en el segundo cuatrimestre de 2024, pensado principalmente para uso personal y orientado a practicar para el final

Fecha de compilación: 11 / 02 / 2025.

Este resumen esta basado en la bibliografía y clases de la materia dictadas en el segundo cuatrimestre de 2024, así como también clases de cuatrimestres pasados de las materias (**F**) de Teoría de Lenguajes y Lógica y Computabilidad

El template de este resumen es esencialmente el mismo usado por @valn y como soy un sinvergüenza lo pedí prestado sin avisar. Si quieren usar un resumen como guía de estudio, recomiendo que usen el suyo antes que el mío.

Link al resumen original:
(https://gitlab.com/valn/uba/-/tree/main/Lenguajes%20Formales%2C%20Aut%C3%B3matas%20y%20Computabilidad/Final?ref_type=heads°)

Índice

1. Lenguajes y Gramáticas

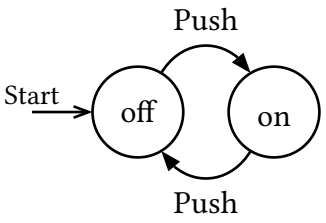
1.1. Por qué nos interesa la teoría de autómatas?

Los autómatas finitos son un modelo útil para varios tipos de software y hardware. Por ejemplo:

- Software para la verificación del comportamiento de circuitos digitales, así como de sistemas con una cantidad finita de estados (protocolos de comunicación, entre otros)
- Los analizadores léxicos de los compiladores
- Software para escanear grandes cuerpos de texto

1.2. Pero entonces, qué son autómatas? (informalmente)

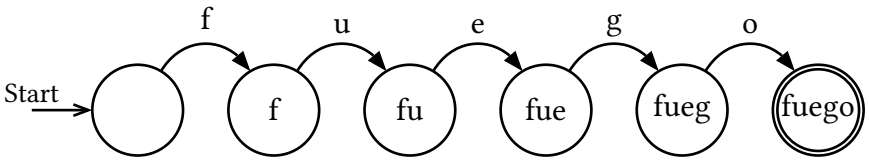
Antes de discutir la definición formal de estos modelos, consideremos primero como un autómata finito se ve y qué es lo que hace.



Acá tenemos un autómata que simula un interruptor de encendido / apagado. el dispositivo recuerda si está en estado «prendido» o «apagado», y permite presionar un botón cuyo efecto va a ser diferente dependiendo del estado en el que se encuentre en ese momento (representado por los arcos saliendo de los estados). Notar la etiqueta «Start», la cual indica el estado inicial del autómata

Hay muchos sistemas que pueden ser vistos de una manera similar, es decir, que cumplen que en cualquier momento determinado se encuentran en uno de un número finito de estados. El propósito de estos estados es recordar la porción relevante de la historia del sistema, para poder actuar de acuerdo a ésta.

Otro ejemplo de un autómata puede ser un analizador léxico. Por ejemplo, un autómata que sólo reconozca la palabra «fuego» podría estar dado por:



Como queremos que sea capaz de reconocer la palabra «fuego», el autómata precisa de 5 estados, cada uno representando una posición diferente de la palabra que ya fue alcanzada. Y los arcos representan un input de una letra Notar ahora el uso de un estado con doble círculos, el mismo denota un estado final, un estado especial que determina que el autómata aceptó el input dado

1.3. Los Conceptos Centrales de la Teoría de Autómatas

Ahora vamos a introducir definiciones esenciales para el estudio de autómatas: el **alfabeto**, las **cadenas**, y el **lenguaje**

1.3.1. Alfabetos

Definición 1.3.1.1: Un *alfabeto* es conjunto finito no vacío de símbolos, que por convención denotamos Σ.

Ejemplo: $\Sigma = \{0, 1\}$. El alfabeto binario

Ejemplo: $\Sigma = \{a, b, ..., z\}$. El alfabeto de todas las letras minúsculas

1.3.2. Cadenas

Definición 1.3.2.1: Una *cadena* (también conocida como palabra) es una secuencia finita de símbolos pertenecientes a un mismo alfabeto

Ejemplo: La cadena 0110 formada por símbolos pertenecientes al alfabeto binario

Definición 1.3.2.2: Nos referimos con *longitud* de una cadena a la cantidad de símbolos en la misma, y denotamos la longitud de la cadena w usando |w|

1.3.2.1. Potencias de un alfabeto

Definición 1.3.2.1.1: Si Σ es un alfabeto, podemos denotar al conjunto de todas las cadenas pertenecientes al alfabeto de cierto longitud usando notación exponencial. Definimos Σ^k como el conjunto de cadenas de longitud k, con todos sus símbolos pertenecientes a Σ

Ejemplo: Para el alfabeto binario, $\Sigma^1 = \{0, 1\}, \Sigma^2 = \{00, 01, 10, 11\}$

Ejemplo: Notar que $\Sigma^0 = \{\lambda\}$, sin importar el alfabeto. Es la única cadena de longitud 0

Definición 1.3.2.1.2: Usamos la notación Σ^* para referirnos al conjunto de todas las cadenas de un alfabeto, y lo denominamos **clausura de Kleene**, y usamos Σ^+ para referirnos a su clausura positiva (es decir, que no incluya la palabra vacía), se definen como:

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i \tag{1}$$

$$\Sigma^+ = \bigcup_{i \geq 1} \Sigma^i \tag{2}$$

1.3.3. Hay tantas palabras como números naturales

Teorema 1.3.3.1: $|\Sigma^*|$ es igual a la cardinalidad de $\mathbb{N}$

Definición 1.3.3.1: Definimos el orden lexicográfico $\prec \subset \Sigma^* \times \Sigma^*$. Asumimos el orden lexicográfico entre los elementos de un mimso alfabeto, y luego lo extendemos a un orden lexicográfico entre palabras de una misma longitud. De esta manera, las palabras de menor longitud son menores en este sentido que las de mayor longitud

Demostración: Definimos una biyección $f : \mathbb{N} \to \Sigma^*$ tal que $f(i) =$ la i-ésima palabra en el orden $\prec$ sobre Σ^*. Luego, como tenemos una biyección entre los conjuntos Σ^* y $\mathbb{N}$, tenemos que necesariamente deben tener la misma cardinalidad □

1.4. Lenguajes

Definición 1.4.1: Un lenguaje L sobre un alfabeto Σ es un conjunto de palabras pertenecientes a Σ. Es decir, $L \subseteq \Sigma^*$

Ejemplo:
- $\emptyset$, el lenguaje vacío, es un lenguaje para cualquier alfabeto
- $\{\lambda\}$, el lenguaje que consiste sólo de la cadena vacía, que también es un lenguaje sobre cualquier alfabeto. (Notar que $\emptyset \neq \{\lambda\}$)
- El lenguaje de todas las cadenas formadas por n 1s seguidos por n 0s sobre $\sigma = \{0, 1\}$. Algunas cadenas de este lenguaje son: {λ, 10, 1100, 111000, ...}

La única restricción importante en cuanto qué puede ser un lenguaje, es que el alfabeto del mismo siempre es finito

Definición 1.4.2: Sea A un conjunto, $\mathcal{P}(A)$ es el conjunto de todos los subconjuntos de A, $\mathcal{P}(A) = \{B \subseteq A\}$. Si tenemos que A es un conjunto finito, entonces $\mathcal{P}(A) = 2^{|A|}$

Teorema 1.4.1: $|\Sigma^*| < \mathcal{P}(\Sigma^*)$, es decir, la cantidad de lenguajes sobre un alfabeto Σ no es numerable

Demostración: Supongamos que $\mathcal{P}(\Sigma^*)$ es numerable, entonces tenemos que podemos enlistar los lengajes y para cada lenguaje L_i (es decir cada lenguaje $\subset \mathcal{P}(\Sigma^*)$), podemos ordenar las palabras pertenecientes al mismo según el orden lexicográfico definido anteriormente de la siguiente manera:
L_1: $w_{1,1}, w_{1,2}, w_{1,3}, ...$
L_2: $w_{2,1}, w_{2,2}, w_{2,3}, ...$
...
Ahora consideremos el lenguaje $L = \{u_1, u_2, u_3, ...\}$ tal que, para todo i, se cumpla que $w_{i,i} \prec u_i$. Pero entonces tenemos un lenguaje cuyo iésimo elemento de L siempre es «mayor» que el iésimo elemento de L_i, entonces no puede ser que L pertenezca a nuestra lista de lenguajes. Pero entonces tenemos un lenguaje sobre Σ^* que no estaba incluido en nuestra ennumeración de todos los lenguajes, por lo que tenemos un absurdo. El mismo vino de suponer que $\mathcal{P}(\Sigma^*)$ era numerable.
□

1.5. Gramáticas

Definición 1.5.1: Una gramática es una 4-upla G = $\langle V_N, V_T, P, S\rangle$ donde:
- V_N es un conjunto de símbolos llamados no terminales
- V_T es un conjunto de símbolos terminales
- P es el conjunto de producciones, que es un conjunto finito de

$$\left[(V_N \cup V_T)^* \ V_N \ (V_N \cup V_T)^*\right] \times (V_N \cup V_T)^*$$

  donde las producciones son tuplas (a, b) y usualmente las notamos $a \to b$
- S es el símbolo distinguido o inicial de V_N

Ejemplo: Sea $G_{\text{arithm}} = \langle V_N, V_T, P, S\rangle$ una gramática libre de contexto (esto último lo introducimos más tarde) tal que:
- $V_N = \{S\}$
- $V_T = \{+, *, a, (,)\}$
- Y producciones determinadas por:
  $S \to S + S$,
  $S \to S * S$,
  $S \to a$,
  $S \to (S)$

1.5.1. Lenguaje generado por una gramática

Definición 1.5.1.1: Dada una gramática G = $\langle V_N, V_T, P, S\rangle$, definimos a $\mathcal{L}(G)$ como:

$$\mathcal{L}(G) = \left\{w \ \in V_T^* : S \overset{+}{\underset{G}{\Rightarrow}} w\right\}$$

donde $\overset{+}{\underset{G}{\Rightarrow}}$ es la derivación en uno o más pasos, que se obtiene de la clausura transitiva de $\underset{G}{\Rightarrow}$ (la derivación directa, cuya definición se da en un momento)

1.5.2. Forma sentencial de una gramática

Definición 1.5.2.1: Sea G = $\langle V_N, V_T, P, S\rangle$.
- S es una forma sentencia de G
- Si $\alpha\beta\gamma$ es una forma sentencial de G, y tenemos que $(\beta \to \delta) \in P$, entonces $\alpha\delta\gamma$ es también una forma sentencial de G

Las formas sentenciales pertenecen a $(V_N \cup V_T)^*$

1.5.3. Derivación directa de una gramática

Definición 1.5.3.1: Si $\alpha\beta\gamma \in (V_N \cup V_T)^*$ y $(\beta \to \delta) \in P$, entonces $\alpha\beta\gamma$ deriva directamente en G a $\alpha\delta\gamma$ y lo denotamos:

$$\alpha\beta\gamma \underset{G}{\Rightarrow} \alpha\delta\gamma$$

Observemos que como la derivación directa es una relación sobre $(V_N \cup V_T)$, podemos componerla consigo misma 0 o más veces:

- $\left(\underset{G}{\Rightarrow}\right)^0 = \mathrm{id}_{(V_N \cup V_T)^*}$
- $\left(\underset{G}{\Rightarrow}\right)^+ = \bigcup_{k=1}^{\infty} \left(\underset{G}{\Rightarrow}\right)^k$
- $\left(\underset{G}{\Rightarrow}\right)^* = \left(\underset{G}{\Rightarrow}\right)^+ \cup \mathrm{id}_{(V_N \cup V_T)^*}$

Según como elijamos derivar, podemos considerar:

- Derivación más a la izquierda: $\underset{L}{\Rightarrow}$
- Derivación más a la derecha: $\underset{R}{\Rightarrow}$

Así como también sus derivaciones en uno o más pasos, y sus calusuras transitivas y de Kleene

1.5.4. La Jerarquía de Chomsky

Definición 1.5.4.1: La jerarquía de Chomsky clasifica las gramáticas en 4 tipos en una jerarquía tal que puedan generar lenguajes cada vez más complejos, y donde cada uno de las gramáticas puede generar también los lenguajes de una de complejidad inferior. Las clasificaciones son:

- **Gramáticas de tipo 0 (o sin restricciones)**:
  $\alpha \to \beta$
- **Gramáticas de tipo 1 (o sensibles al contexto)**:

$\alpha \to \beta$, con $|\alpha| \leq |\beta|$

- **Gramáticas de tipo 2 (o libres de contexto)**:

A $\to \gamma$ con A $\in V_N$

- **Gramáticas de tipo 3 (o regulares)**:

$A \to a, A \to aB, A \to \lambda$, con $A, B \in V_N, a \in V_T$

La jerarquía de grmáticas da origen también a la jerarquía de lenguajes:

- **Recursivamente enumerables**
- **Sensitivos al contexto**
- **Libres de contexto**
- **Regulares**

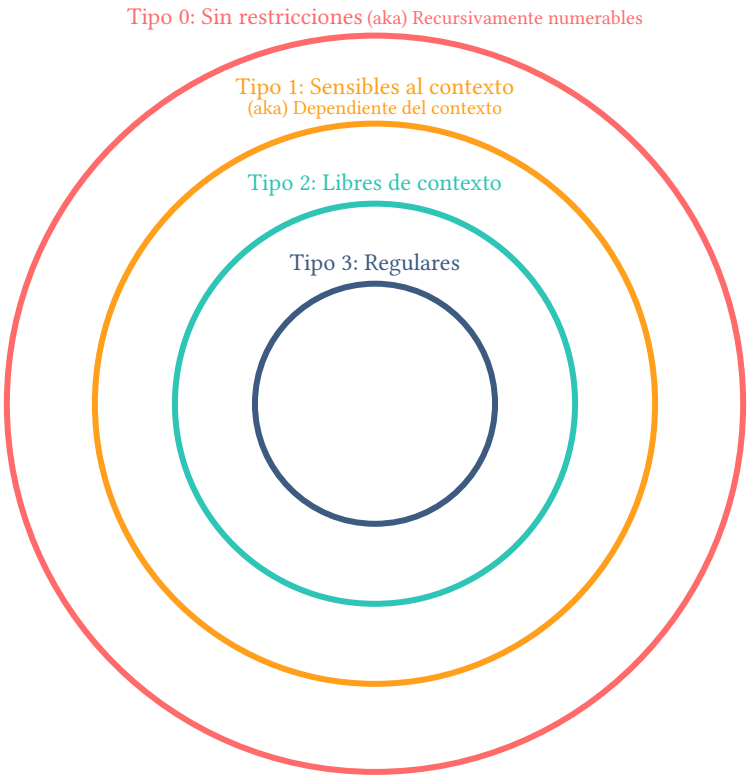


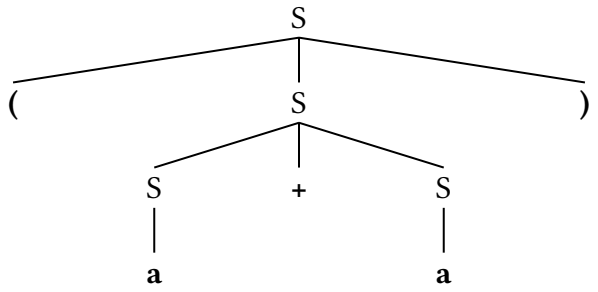
Figura 1: Imagen perteneciente al repo de @Valn (No sé que haces acá todavía a ser honesto).

1.5.5. Árbol de derivación de gramáticas

Definición 1.5.5.1: Un árbol de derivación es una representación gráfica de una derivación (look at me I'm so smart) donde:

- Las etiquetas de las hojas están en $V_T \cup \{\lambda\}$
- Las etiquetas de los nodos internos están en V_N. Las etiquetas de sus símbolos son los símbolos del cuerpo de una producción
- Un nodo tiene etiqueta A y tiene n descendientes etiquetados $X_1, X_2, ..., X_N$ sii hay una derivación que usa una producción $A \to X_1 X_2 ... X_N$

Ejemplo: Considerande la gramática G_{arithm} definida anteriormente, un posible árbol de derivación podría ser el siguiente:

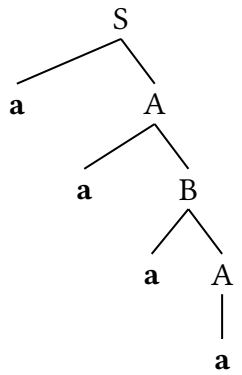


$$S \underset{L}{\Rightarrow} (S) \underset{L}{\Rightarrow} (S+S) \underset{L}{\Rightarrow} (a+S) \underset{L}{\Rightarrow} (a+S) \underset{L}{\Rightarrow} (a+a)$$

1.5.6. Algunos lemas relevantes

Lema 1.5.6.1: Sea $G = \langle V_N, V_T, P, S \rangle$ regular, no recursiva a izquierda. Si $A \overset{i}{\underset{L}{\Rightarrow}} wB$ entonces $i = |w|$

Demostración: Consideremos el árbol de derivación de la cadena w = $a_1 a_2 a_3 ... a_n$. Si entonces cortamos el árbol en una altura h determinada, para $h \geq 1$ se obtiene un subarbol de h hojas, con el único nodo que no es hoja con una etiqueta de V_N. Por la forma que las producciones de una gramática regular tiene, tenemos que cada derivación pone un símbolo no terminal a lo sumo y uno terminal , luego, tenemos que en n derivaciones (altura n del árbol) tenemos n hojas y el no terminal de la derecha □



Un posible árbol de derivación para las producciones:

$$S \to aA \mid \lambda$$
$$A \to a \mid aB$$
$$B \to aA$$

Lema 1.5.6.2: Sea G = $\langle V_N, V_T, P, S\rangle$ libre de contexto, no recursiva a izquierda (es decir, no tiene derivaciones $A \underset{L}{\overset{+}{\Rightarrow}} A\alpha$). Existe una constante c tal que si $A \underset{L}{\overset{i}{\Rightarrow}} \omega B\alpha$ entonces $i \leq c^{|\omega|+2}$

Demostración: Pendiente (soy ***vago***) □

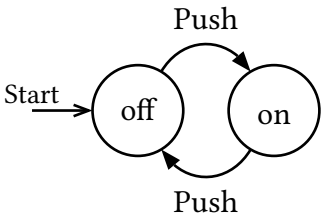
2. Autómatas Finitos Determinísticos, no Determinísticos, y Gramáticas Regulares

2.1. Autómata Finito Determinístico (AFD)

Definición 2.1.1: Un autómata finito determinístico es una 5-upla $\langle Q, \Sigma, \delta, q_0, F \rangle$ donde:
- Q es un conjunto finito de estados
- Σ es el alfabeto de entrada del autómata
- δ es la función de trancición del autómata que toma como argumentos un estado y un símbolo de input, y devuelve un estado, es decir: $\delta : Q \times \Sigma \to Q$.
- $q_0 \in Q$ es el estado inicial del autómata
- $F \subseteq Q$ es el conjunto de estados finales del autómata

Para determinar si un autómata acepta cierta cadena, se hace uso de su función de transición y se confirma si se termina llegando a un estado $q_f \in F$ final. Es decir, suponiendo que se busca. Como es poco práctico escribir la función δ en su totalidad, se suele hacer uso de diagramas de autómatas (como el usado al principio de este resumen) y se aceptan los arcos del mismo como una definición implícita.

Ejemplo: En nuestro primer autómata, teníamos (notar que «Push» es sólo un símbolo, no una cadena / palabra)



Luego, la 5-upla correspondiente sería $\langle \{\text{off}, \text{on}\}, \{\text{Push}\}, \delta, \text{off}, \emptyset \}$ con δ definido por:
$\delta(\text{off}, \text{Push}) = \text{on}$
$\delta(\text{on}, \text{Push}) = \text{off}$

2.1.1. Función de transición generalizada $\hat{\delta}$

Ahora necesitamos definir precisamente qué ocurre cuando estamos en un estado determinado y recibimos como input una cadena, no un símbolo. Para esto, lo definimos por inducción en el largo de la cadena

Definición 2.1.1.1: Definimos $\hat{\delta}$ como:
- $\hat{\delta}(q, \lambda) = q$
- $\hat{\delta}(q, \omega\alpha) = \delta\Big(\hat{\delta}(q, \omega), \alpha\Big)$ con $\omega \in \Sigma^* y \alpha \in \Sigma$

Cabe recalcar que $\hat{\delta}(q, \alpha) = \delta\Big(\hat{\delta}(q, \lambda), \alpha\Big) = \delta(q, \alpha)$ y que muchas veces vamos a hacer uso de la misma notación para ambas funciones

2.1.2. El Lenguaje de un AFD

Definición 2.1.2.1: Ahora podemos definir el lenguaje aceptado por un AFD como:

$$\mathcal{L}(M) = \left\{\omega \in \Sigma^* : \hat{\delta}(q_0, \omega) \in F\right\}$$

Es decir, el lenguaje de un autómata M es el conjunto de cadenas que mediante $\hat{\delta}$ llegan desde el estado inicial q_0 a un estado final. Llamamos **lenguajes regulares** a aquellos aceptados por un AFD

2.2. Autómatas Finitos no Determinísticos (AFND)

Definición 2.2.1: Un AFND es una 5-upla $\langle Q, \Sigma, \delta, q_0, F\rangle$ donde:

- Q es un conjunto finito de estados
- Σ es el alfabeto de entrada
- δ es la función de trancición, que toma un estado en Q y un símbolo en Σ, pero que ahora devuelve un subconjunto de Q. Es decir:

$$\delta : Q \times \Sigma \to \mathcal{P}(Q)$$

- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales

2.2.1. Función de trancisión generalizada $\hat{\delta}$

Definición 2.2.1.1: Definimos $\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$:

- $\hat{\delta}(q, \lambda) = \{q\}$
- $\hat{\delta}(q, x\alpha) = \left\{p \in Q : \exists r \in \hat{\delta}(q, x) \;\; y \;\; p \in \delta(r, a)\right\}$
- Una definición alternativa, siendo $w = xa$, con $w, x \in \Sigma^* y \;\; a \in \Sigma$ y suponiendo que $\hat{\delta}(q, x) = \{p_1, ..., p_k\}$. Sea

  $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, ..., r_k\}$

  Entonces tenemos $\hat{\delta}(q, w) = \{r_1, ..., r_k\}$

Notar que:

$$\begin{aligned}\hat{\delta}(q, \lambda\alpha) &= \left\{p \in Q : \exists r \in \hat{\delta}(q, \lambda) \;\; y \;\; p \in \delta(r, a)\right\} \\ &= \{p \in Q : \exists r \in \{q\} \;\; y \;\; p \in \delta(r, a)\} \\ &= \{p \in Q : p \in \delta(q, a)\} \\ &= \delta(q, a)\end{aligned}$$

2.2.2. Lenguaje aceptado por un AFND

Definición 2.2.2.1: El lenguaje aceptado por un AFND M, $\mathcal{L}(M)$, es el conjunto de cadenas aceptadas por M y está definido como:

$$\mathcal{L}(M) = \left\{x \in \Sigma^* : \hat{\delta}(q_0, x) \cap F \neq \emptyset\right\}$$

2.2.3. Función de transición de conjuntos de estados

Definición 2.2.3.1: Podemos extender la función de transición aún más, haciendo que mapee conjuntos de estados y cadenas en conjuntos de estados. Sea entonces $\delta : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$ dada por:

$$\delta(P, x) = \bigcup_{q \in P} \hat{\delta}(q, x)$$

2.3. Equivalencia Entre AFND y AFD

Teorema 2.3.1: Dado un AFND $N = \langle Q_N, \Sigma, \delta_N, q_0, F_N\rangle$, existe un AFD $D = \langle Q_D, \Sigma, \delta_D, \{q_0\}, F_D\rangle$ tal que $\mathcal{L}(N) = \mathcal{L}(D)$

Demostración: La demostración comienza mediante una construcción llamada *construcción de subconjunto*, llamada así porque construye un autómata a partir de subconjuntos del conjunto de estados de otro (es decir, subconjuntos de $\mathcal{P}(Q)$).
Dado el autómata N AFND, construimos al autómata D de la siguiente manera:

- Q_D es el conjunto de subconjuntos de Q_N (es decir, $\mathcal{P}(Q_N)$). Notar que si Q_N tenía n estados, Q_D va a tener 2^n estados (ouch)
- F_D es el conjunto de subconjuntos S de Q_N tal que $S \cap F_N \neq \emptyset$
- Para cada S $\subseteq Q_N$ y símbolo $\alpha \in \Sigma$:

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

Ahora, probamos primero por inducción sobre $|\omega|$ que vale:

$$\hat{\delta}_D(\{q_0\}, \omega) = \hat{\delta}_N(q_0, \omega)$$

Notar que ambas $\hat{\delta}$ devuelven un conjunto de estados de Q_N, pero para nuestro AFD D el resultado es interpretado como un estado, mientras que en el AFND N se trata de un subconjunto de estados de Q_N.

- Caso Base: $|\omega|$ = 0 (O sea, $\omega = \lambda$)
  Por la definición de ambos $\hat{\delta}$ tenemos que ambos son $\{q_0\}$

- Paso Inductivo: Sea ω delongitud n + 1 y asumiendo que la H.I vale para n. Separamos ω = $x\alpha$, donde α es el último símbolo de ω. Por la H.I, tenemos que $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x) = \{p_1, ..., p_k\}$. conjuntos de estados $\in Q_N$.

Recordando la definición de AFND teníamos que:

$$\hat{\delta}_N(q_0, \omega) = \bigcup_{i=1}^{k} \delta_N(p_i, \alpha)$$

Por otro lado, al construir el AFD definimos que:

$$\delta_D(\{p_1, p_2, ..., p_k\}, \alpha) = \bigcup_{i=1}^{k} \delta_N(p_i, \alpha)$$

Con esto en mente, podemos resolver:

$$\hat{\delta}_D(\{q_0\}, \omega) = \delta_D\Big(\hat{\delta}_D(\{q_0\}, x), \alpha\Big) \underset{HI}{=} \delta_D(\{p_1, ..., p_k\}, \alpha) = \bigcup_{i=1}^{k} \delta_N(p_i, \alpha) = \hat{\delta}_N(q_0, \omega)$$

Sabiendo ahora que ambas funciones de transición son "equivalentes" y que ambas aceptan una cadena sii el conjunto resultante contiene un estado final, tenemos que $\mathcal{L}(N) = \mathcal{L}(D)$ □

2.4. AFND con transiciones λ $(AFND - \lambda)$

Definición 2.4.1: Un AFND-λ es una 5-upla $\langle Q, \Sigma, \delta, q_0, F\rangle$ donde todos los componentes tienen la misma interpretación que antes con la excepción de que δ ahora tiene su dominio definido como:

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$$

Definición 2.4.2: Definimos como **clausura $\boldsymbol{\lambda}$** de un estado q, denotado $Cl_\lambda(q)$, al conjunto de estados alcanzables desde q mediante transiciones λ. Es decir, sea R $\subseteq Q \times Q$ tal que (q, p) $\in R \iff p \in \delta(q, \lambda)$. Definimos $Cl_\lambda : Q \to \mathcal{P}(Q)$ como:

$$Cl_\lambda(q) = \{p : (q, p) \in R^*\}$$

Definición 2.4.3: Podemos también extender la definición para un conjunto de estados P:

$$Cl_\lambda(P) = \bigcup_{q \in P} Cl_\lambda(q)$$

Definición 2.4.4: La función de trancisión δ puede extenderse para acept ar cadenas en Σ, es decir, $\hat{\delta} : Q \times \Sigma^* \to P(Q)$, y la definimos de la siguiente manera:

- $\hat{\delta}(q, \lambda) = Cl_\lambda(q)$.

- $\hat{\delta}(q, x\alpha) = Cl_\lambda\Big(\big\{p : \exists r \in \hat{\delta}(q, x) : p \in \delta(r, \alpha)\big\}\Big)$ con x $\in \Sigma^*$ y $\alpha \in \Sigma$ o, lo que es equivalente
$$\hat{\delta}(q, x\alpha) = Cl_\lambda\Big(\bigcup_{r \in \hat{\delta}(q,x)} \delta(r, \alpha)\Big)$$

Extendiendo la definición a conjunto de estados, tenemos que:

- $\delta(P, \alpha) = \bigcup_{q \in P} \delta(q, \alpha)$
- $\hat{\delta}(P, x) = \bigcup_{q \in P} \hat{\delta}(q, x)$

Lo que nos permite reescribir $\hat{\delta}(q, x\alpha)$ como:
$$\hat{\delta}(q, x\alpha) = Cl_\lambda\Big(\delta\Big(\hat{\delta}(q, x), \alpha\Big)\Big)$$

Definición 2.4.5: Definimos al lenguaje aceptado por un AFND-λ E:
$$\mathcal{L}(E) = \Big\{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\Big\}$$

Teorema 2.4.1: Dado un AFND-λ E = $\langle Q, \Sigma, \delta_\lambda, q_0, F_\lambda\rangle$ puede encontrarse un AFND N = $\langle Q, \Sigma, \delta_N, q_0, F_N\rangle$ que reconoce el mismo lenguaje

Demostración: Tomemos
$$F_N = \begin{cases} F_\lambda \cup \{q_0\} \text{ si } \ Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \\ F_\lambda \text{ si no} \end{cases}$$

Y tomemos $\hat{\delta}_N(q, \alpha) = \hat{\delta}_\lambda(q, \alpha)$. Para empezar, vamos a probar que $\hat{\delta}_N(q, x) = \hat{\delta}_\lambda(q, x)$ para $|x| \geq 1$. Lo hacemos por inducción:

- Para |x| = 1:
  $x = \alpha$, es decir, un símbolo del alfabeto, por lo que se cumple por definición
- Para |x| >:
  Tomemos $x = \omega\alpha$ entonces:
$$\hat{\delta}_N(q_0, \omega\alpha) = \hat{\delta}_N\Big(\hat{\delta}_N(q_0, \omega), \alpha\Big) \underset{H.I}{=} \hat{\delta}_N\Big(\hat{\delta}_\lambda(q_0, \omega), \alpha\Big)$$

  Por otro lado, tenemos que para $P \subseteq Q$ es cierto que $\hat{\delta}_N(P, \alpha) = \hat{\delta}_\lambda(P, \alpha)$ ya que:

$$\hat{\delta}_N(P,\alpha) = \bigcup_{q\in P} \hat{\delta}_N(q,\alpha) = \bigcup_{q\in P} \hat{\delta}_\lambda(q,\alpha) = \hat{\delta}_\lambda(P,\alpha)$$

Luego, con $P = \hat{\delta}_\lambda(q_0,\omega)$ tenemos que ambas funciones de transición son "equivalentes" pues:

$$\hat{\delta}_N(q_0,\omega) = \hat{\delta}_N\Big(\hat{\delta}_\lambda(q_0,\omega),\alpha\Big) = \hat{\delta}_\lambda\Big(\hat{\delta}_\lambda(q_0,\omega),\alpha\Big) = \hat{\delta}_\lambda(q_0,\omega\alpha)$$

Ahora queda ver que $\mathcal{L}(N) = \mathcal{L}(E)$. Para $x = \lambda$ tenemos

- $\lambda \in \mathcal{L}(E) \iff \hat{\delta}_\lambda(q_0,\lambda) \cap F_\lambda \neq \emptyset \underset{\hat{\delta}_\lambda(q_0,\lambda)=Cl_\lambda(q_0)}{\iff} Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset$
$$\implies q_0 \in F_N \iff \lambda \in \mathcal{L}(N)$$

- $\lambda \in \mathcal{L}(N) \iff q_0 \in F_N \implies q_0 \in F_\lambda \vee Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \iff$
$Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \vee \lambda \in \mathcal{L}(E) \iff \lambda \in \mathcal{L}(E) \vee \lambda \in \mathcal{L}(E) \iff \lambda \in \mathcal{L}(E)$

Para |x| $\neq \lambda$:

- $x \in \mathcal{L}(E) \underset{\text{AFND-}\lambda}{\overset{\text{def cadena aceptada}}{\iff}} \hat{\delta}_\lambda(q_0,x) \cap F_\lambda \neq \emptyset \underset{\hat{\delta}_\lambda(q_0,x)=\hat{\delta}_N(q_0,x)}{\overset{\text{porque } F_\lambda \subset F_N}{\implies}}$
$$\hat{\delta}_N(q_0,x) \cap F_N \neq \emptyset \implies x \in \mathcal{L}(N)$$

Por el otro lado:

- $x \in \mathcal{L}(N) \iff \hat{\delta}_N(q_0,x) \cap F_N \neq \emptyset$
$$\implies \hat{\delta}_\lambda(q_0,x) \cap F_\lambda \neq \emptyset \vee \Big(\hat{\delta}_\lambda(q_0,x) \cap \{q_0\} \neq \emptyset \wedge Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset\Big)$$

- Del lado izquierdo de la disyunción tenemos:
$$\hat{\delta}_\lambda(q_0,x) \cap F_\lambda \neq \emptyset \implies x \in \mathcal{L}(E)$$

- Del lado derecho tenemos que:
  - $\hat{\delta}_\lambda(q_0,x) \cap \{q_0\} \neq \emptyset \implies$ existe un loop x de q_0 sobre sí mismo
  - $Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset \implies$ existe un camino λ desde q_0 hasta F_λ, por lo que existe un camino x desde q_0 hasta F_λ

- Juntando los dos, tenemos que:
$$\Big(\hat{\delta}_\lambda(q_0,x) \cap \{q_0\} \neq \emptyset \wedge Cl_\lambda(q_0) \cap F_\lambda \neq \emptyset\Big) \implies x \in \mathcal{L}(E)$$

Finalmente, podemos concluir que:

$$x \in \mathcal{L}(E) \iff x \in \mathcal{L}(N)$$

□

2.5. Gramáticas regulares y AFDs

Recordemos que una gramática G = $\langle V_N, V_T, P, S\rangle$ es regular si todas sus producciones son de la forma $A \to \lambda$, $A \to a$ o $A \to aB$

Teorema 2.5.1: Dada una gramática regular G = $\langle V_N, V_T, P, S\rangle$, existe un AFND $N = \langle Q, \Sigma, \delta, q_0, F\rangle$ tal que $\mathcal{L}(G) = \mathcal{L}(M)$

Demostración: Definamos N de la siguiente manera:

- $Q = V_N \cup \{q_f\}$. A partir de ahora, usamos q_A para referirnos al estado correspondiente al no terminal A
- $\Sigma = V_T$
- $q_0 = q_S$
- $q_B \in \delta(q_A, a) \iff A \to aB \in P$
- $q_f \in \delta(q_A, a) \iff A \to a \in P$
- $q_A \in F \iff A \to \lambda \in P$
- $q_f \in F$

Como paso intermedio, ahora probamos el siguiente lema:

Lema 2.5.2: Para todo $\omega \in V_T^*$, si $A \overset{*}{\to} \omega B$ entonces $q_B \in \hat{\delta}(q_A, \omega)$

Demostración: Por inducción en la longitud de ω

- Caso base $|\omega| = 0$, $(\omega = \lambda)$

Como $A \overset{*}{\to} A$ y $q_a \in \hat{\delta}(q_A, \lambda)$ tenemos por definición de N que $A \overset{*}{\to} A \iff q_a \in \hat{\delta}(q_A, \lambda)$

- Caso $|\omega| = n + 1, n \geq 0$, con, $\omega = x\alpha$:

$$
\begin{aligned}
A \overset{*}{\to} x\alpha B &\iff \exists C \in V_N : A \overset{*}{\to} xC \wedge C \to \alpha B \in P \\
&\underset{HI}{\Longrightarrow} \exists q_C \in Q : q_C \in \hat{\delta}(q_A, x) \wedge q_B \in \delta(q_C, a) \\
&\iff q_B \in \delta\Big(\hat{\delta}(q_A, x), a\Big) \\
&\iff q_B \in \hat{\delta}(q_A, x\alpha)
\end{aligned}
$$

□

Volviendo ahora con el teorema,

$$\omega\alpha \in \mathcal{L}(G) \underset{\substack{\text{def lenguaje generado}\\ \text{por una gramática}}}{\iff} S \overset{*}{\to} \omega\alpha$$

$$\iff \Big(\exists A \in V_N, S \overset{*}{\to} \omega A \land A \to \alpha \in P\Big) \lor \Big(\exists B \in V_N, S \overset{*}{\to} \omega\alpha B \land B \to \lambda\Big)$$

$$\iff \Big(\exists A \in V_N, S \overset{*}{\to} \omega A \land q_f \in \delta(q_A, \alpha)\Big) \lor \Big(\exists B \in V_N, S \overset{*}{\to} \omega\alpha B \land q_B \in F\Big)$$

$$\underset{\text{por lema}}{\Longrightarrow} \Big(\exists q_A \in Q : q_A \in \hat{\delta}(q_S, \omega) \land q_f \in \delta(q_A, \alpha)\Big) \lor \Big(\exists q_B \in Q : q_B \in \hat{\delta}(q_S, \omega\alpha) \land q_B \in F\Big)$$

$$\iff \Big(q_f \in \hat{\delta}(q_S, \omega\alpha)\Big) \lor \Big(\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset\Big)$$

$$\iff \Big(\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset\Big) \lor \Big(\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset\Big)$$

$$\iff \Big(\hat{\delta}(q_S, \omega\alpha) \cap F \neq \emptyset\Big)$$

$$\iff \omega\alpha \in \mathcal{L}(N)$$

Luego, queda demostrado que $\mathcal{L}(G) = \mathcal{L}(N)$

Teorema 2.5.3: Dado un AFD $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ existe una gramática regular $G = \langle V_N, V_T, P, S\rangle$ tal que $\mathcal{L}(M) = \mathcal{L}(G)$

Demostración: Para comenzar definimos una gramática $G = \langle V_N, V_T, P, S\rangle$, donde $V_N = Q$ y llamaremos A_p al no terminal correspondiente a p $\in Q$; $S = A_{q_0}$; $V_T = \Sigma$ y el conjunto P es:

$$A_p \to aA_q \in P \iff \delta(p, a) = q$$
$$A_p \to a \in P \iff \delta(p, a) = q \in F$$
$$S \to \lambda \in P \iff q_0 \in F$$

Ahora vamos a probar el siguiente lema como paso intermedio:

Lema 2.5.4: $\delta(p, \omega) = q \iff A_p \overset{*}{\to} \omega A_q$

Lo probamos por induccion en la longitud de ω:

Para $\omega = \lambda$ tenemos que $\delta(p, \lambda) = p$ y que $A_p \overset{*}{\to} A_p$, por lo que $\delta(p, \lambda) = p \iff A_p \overset{*}{\to} A_p$

Veamos que vale ahora para $\omega = x\alpha$:

$$\delta(p, x\alpha) = q \iff \exists r \in Q, \hat{\delta}(p, x) = r \wedge \delta(r, \alpha) = q$$
$$\underset{HI}{\iff} \exists A_r, A_p \overset{*}{\to} xA_r \wedge A_r \to \alpha A_q \in P$$
$$\iff A_P \overset{*}{\to} x\alpha A_q$$

Luego, probamos el lema. Volviendo ahora a la demo del teorema:

$$\begin{aligned}\lambda \in \mathcal{L}(M) &\iff q_0 \in F \\ &\iff (S \to \lambda) \in P \\ &\iff S \overset{*}{\Rightarrow} \lambda \\ &\iff \lambda \in \mathcal{L}(G).\end{aligned}$$

Caso cadena no vacía:

$$\begin{aligned}wa \in \mathcal{L}(M) &\iff \delta(q_0, wa) \in F \\ &\iff \exists p \in Q : \delta(q_0, w) = p \wedge \delta(p, a) \in F \\ &\underset{\text{Lema}}{\iff} \exists A_p \in V_N : A_{q_0} \overset{*}{\Rightarrow} wA_p \wedge \left(A_p \to a\right) \in P \\ &\iff A_{q_0} \overset{*}{\Rightarrow} wa \\ &\iff wa \in \mathcal{L}(G).\end{aligned}$$

Con esto queda demostrado que $\mathcal{L}(M) = \mathcal{L}(G)$.

□

3. Expresiones Regulares

Definición 3.1: Una expresión regular es una cadena de símbolos de un alfabeto que denotan un lenguaje sobre el alfabeto. Las expresiones regulares se construyen a partir de los siguientes elementos:

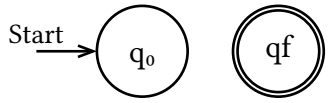
- $\emptyset$ es una expresión regular que representa el lenguaje vacío $\emptyset$
- λ es una expresión regular que representa el lenguaje que contiene sólo la cadena vacía $\{\lambda\}$
- a es una expresión regular que representa el lenguaje que contiene la cadena a para cada $a \in \Sigma$ ($\{a\}$)
- Si r y s son expresiones regulares que denotan los lenguajes R y S, entonces:
  - $(r + s) = (r \mid s)$ es una expresión regular que representa la unión de los lenguajes R y S.
  - (rs) es una expresión regular que representa la concatenación de los lenguajes R y S.
  - (r^*) es una expresión regular que representa la clausura de Kleene del lenguaje representado por r.
  - r^+ representa la clausura positiva del lenguaje representado por r.

Ejemplo: Algunos ejemplos de expresiones regulares son:
- $a + b$ representa el lenguaje que contiene las cadenas a y b
- a^*b representa el lenguaje que contiene las cadenas que comienzan con una cantidad arbitraria de a seguida de una b
- $(a + b)^*$ representa el lenguaje que contiene todas las cadenas que contienen únicamente a y b

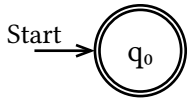
Teorema 3.1: Dada una expresión regular r, existe un AFND-λ con un solo estado final y sin transiciones a partir del mismo $E = \langle Q, \Sigma, \delta, q_0, F\rangle$ tal que $\mathcal{L}(r) = \mathcal{L}(E)$

Demostración: Por inducción sobre la estructura de la expresión regular r:

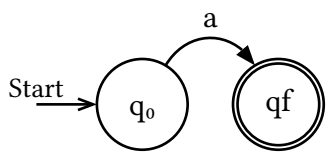
- Caso base: $r = \emptyset$



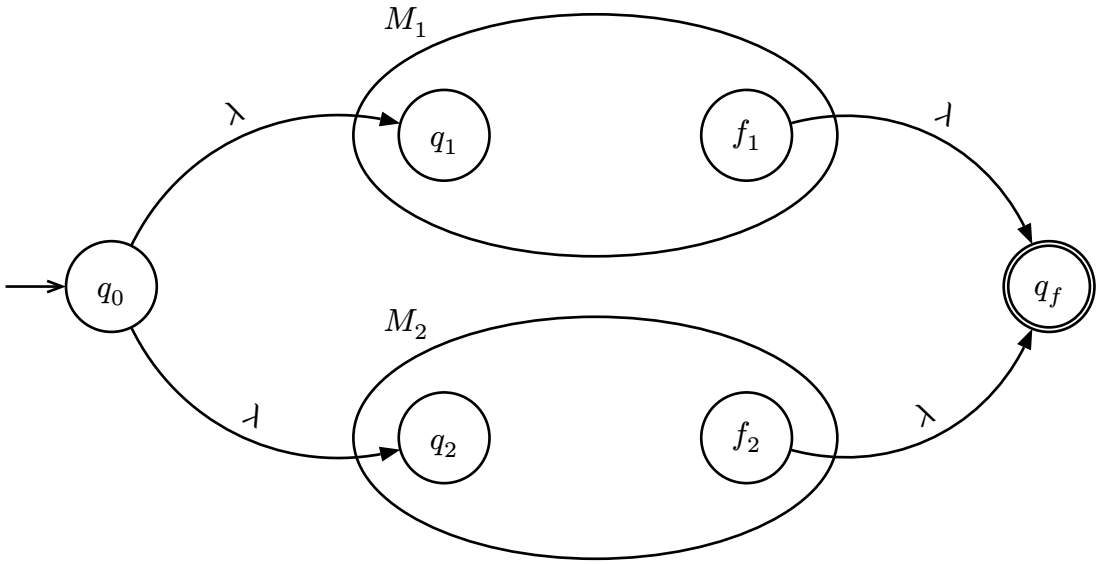
- Caso base: $r = \lambda$



- Caso base: $r = a$



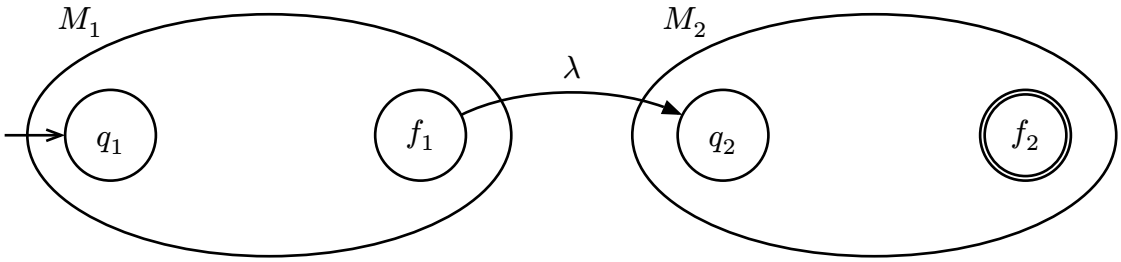
- Caso inductivo: $r = r_1 + r_2$ Por H.I, existen $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, \{f_1\}\rangle$ y $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_2, \{f_2\}\rangle$ tales que $\mathcal{L}(M_1) = \mathcal{L}(r_1)$ y $\mathcal{L}(M_2) = \mathcal{L}(r_2)$. Sea $M = \langle Q_1 \cup Q_2 \cup \{q_0, q_f\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{q_f\}\rangle$:



- $\delta(q_0, \lambda) = \{q_1, q_2\}$.
- $\delta(q, a) = \delta_1(q, a)$ para $q \in Q_1 - \{f_1\}$ y $a \in \Sigma_1 \cup \{\lambda\}$.
- $\delta(q, a) = \delta_2(q, a)$ para $q \in Q_2 - \{f_2\}$ y $a \in \Sigma_2 \cup \{\lambda\}$.
- $\delta(f_1, \lambda) = \delta(f_2, \lambda) = \{q_f\}$.

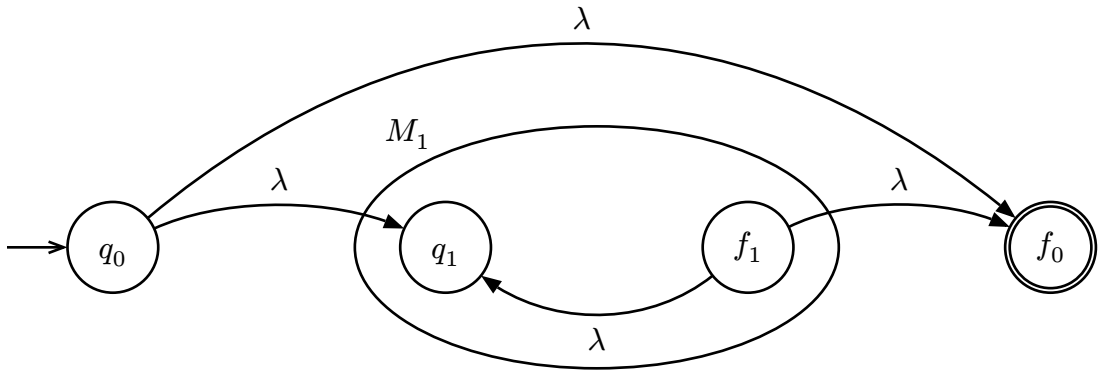
De manera informal, podemos alcanzar, a partir del nuevo estado inicial, cualquiera de los estados iniciales correspondientes a los autómatas para r_1 y r_2 mediante transiciones λ. Luego, podemos seguir las transiciones de los autómatas hasta llegar a sus estados previamente finales correspondientes, y luego tomar una úlima transición λ al nuevo estado final.

- Caso inductivo: $r = r_1 r_2$
  Por HI existen $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, \{f_1\}\rangle$ y $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_2, \{f_2\}\rangle$ tales que $\mathcal{L}(M_1) = \mathcal{L}(r_1)$ y $\mathcal{L}(M_2) = \mathcal{L}(r_2)$.
  Entones sea $M = \langle Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1, \{f_2\}\rangle$:



  - $\delta(q, a) = \delta_1(q, a)$ para $q \in Q_1 - \{f_1\}$ y $a \in \Sigma_1 \cup \{\lambda\}$.
  - $\delta(f_1, \lambda) = \{q_2\}$.
  - $\delta(q, a) = \delta_2(q, a)$ para $q \in Q_2 - \{f_2\}$ y $a \in \Sigma_2 \cup \{\lambda\}$.

- Caso inductivo: $r = r_1^*$
  Por HI existe $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, \{f_1\}\rangle$ tal que $\mathcal{L}(M_1) = \mathcal{L}(r_1)$.
  Entonces, podemos construir el autómata $M = \langle Q_1 \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, \{f_0\}\rangle$.



- $\delta(q, a) = \delta_1(q, a)$ para $q \in Q_1 - \{f_1\}$ y $a \in \Sigma_1 \cup \{\lambda\}$.
- $\delta(q_0, \lambda) = \delta(f_1, \lambda) = \{q_1, f_0\}$.

- Caso infuctivo $r = r_1^+$:

  Dado que $r_1^+ = r_1 r_1^*$, queda demostrado por los casos anteriores.

  Con esto, queda demostrado que $\mathcal{L}(r) = \mathcal{L}(M)$ para todo r expresión regular.

□

Teorema 3.2: Dado un AFD $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, existe una expresión regular r tal que $\mathcal{L}(r) = \mathcal{L}(M)$

Demostración: Como tenemos que los estados de un autómata son finitos, podemos renombrar los mismos como $\{q_1, q_2..., q_n\}$con $n = |Q|$. Con esto en cuenta, denotamos con $R_{i,j}^k$ al conjunto de cadenas que llevan al autómata de q_i a q_j sin pasar por ningún estado intermedio con índice mayor que k. Luego, definimos $R_{i,j}^k$ inductivamente como:

- $R_{i,j}^0 = \begin{cases} \{a : \delta(q_i, a) = q_j\} : \ a \in \Sigma \text{ si } \ i \neq j \\ \{a : \delta(q_i, a) = q_j\} \cup \{\lambda\} \text{ si } \ i = j \end{cases}$
- $R_{i,j}^k = R_{i,j}^{k-1} \cup R_{i,k}^{k-1} \left(R_{k,k}^{k-1}\right)^* R_{k,j}^{k-1}$

Como paso intermedio, queremos demostrar que para todo $R_{i,j}^k$ existe una e.r $r_{i,j}^k$ tal que $\mathcal{L}\left(r_{i,j}^k\right) = R_{i,j}^k$ Haciendo inducción sobre k:

- Caso base: $k = 0$
  $R_{i,j}^0$ es el conjunto de cadenas de un solo caracter o λ. Por lo que la e.r $r_{i,j}^k$ que lo denota será:
  - $\emptyset$ si no existe ningún a_i que una q_i y q_j y i ≠ j
  - λ si no existe ningún a_i que una q_i y q_j, pero con i = j
  - $a_1 \mid ... \mid a_p$ con a_l simbolos del alfabeto, si delta(q_i, a_l) = q_j y j ≠ i
  - $a_1 \mid ... \mid a_p \mid \lambda$ con a_l simbolos del alfabeto, si delta(q_i, a_l) = q_j y j = i
- Paso inductivo: Por H.I tenemos que:

$$\mathcal{L}\left(r_{i,k}^{k-1}\right) = R_{i,k}^{k-1} \quad \mathcal{L}\left(r_{k,k}^{k-1}\right) = R_{k,k}^{k-1} \quad \mathcal{L}\left(r_{k,j}^{k-1}\right) = R_{k,j}^{k-1} \quad \mathcal{L}\left(r_{i,j}^{k-1}\right) = R_{i,j}^{k-1}$$

Si definimos $r_{i,j}^k = r_{i,k}^{k-1} \left(r_{k,k}^{k-1}\right)^* r_{k,j}^{k-1} \mid r_{i,j}^{k-1}$ tenemos que:

$$\mathcal{L}\left(r_{i,j}^{k}\right) =$$

$$\mathcal{L}\left(r_{i,k}^{k-1}\left(r_{k,k}^{k-1}\right)^{*}r_{k,j}^{k-1} \mid r_{i,j}^{k-1}\right) =$$

$$\mathcal{L}\left(r_{i,k}^{k-1}\left(r_{k,k}^{k-1}\right)^{*}r_{k,j}^{k-1}\right) \cup \mathcal{L}\left(r_{i,j}^{k-1}\right) =$$

$$\mathcal{L}\left(r_{i,k}^{k-1}\right)\mathcal{L}\left(\left(r_{k,k}^{k-1}\right)^{*}\right)\mathcal{L}\left(r_{k,j}^{k-1}\right) \cup \mathcal{L}\left(r_{i,j}^{k-1}\right) =$$

$$\mathcal{L}\left(r_{i,k}^{k-1}\right)\left(\mathcal{L}\left(r_{k,k}^{k-1}\right)^{*}\right)\mathcal{L}\left(r_{k,j}^{k-1}\right) \cup \mathcal{L}\left(r_{i,j}^{k-1}\right) =$$

$$R_{i,k}^{k-1}\left(R_{k,k}^{k-1}\right)^{*}R_{k,j}^{k-1} \cup R_{i,j}^{k-1} =$$

$$R_{i,j}^{k}$$

Con esto en mente, notemos primero que el lenguaje denotado por el autómata M está dado por todas las cadenas que llevan al autómata de q_0 a un estado final. Es decir:

$$\mathcal{L}(M) = \bigcup_{q_j \in F} R_{1,j}^{n}$$

Luego, tenemos que:

$$\mathcal{L}(M) = \bigcup_{q_j \in F} \mathcal{L}\left(R_{1,j}^{n}\right) = \bigcup_{q_j \in F} \mathcal{L}\left(r_{1,j}^{n}\right) = \mathcal{L}\left(r_{1,j1}^{n} \mid \ldots \mid r_{1,jm}^{n}\right)$$

Por lo que concluimos que $\mathcal{L}(M)$ es denotado por la e.r $r_{1,j1}^{n} \mid \ldots \mid r_{1,jm}^{n}$ $\square$

4. Lema de Pumping y propiedades de clausura de los lenguajes regulares

4.1. Configuración instantánea de un AFD

Para comenzar, vamos a introducir una nueva notación que nos facilitará un par de pruebas

Definición 4.1.1: Sea AFD M = $\langle Q, \Sigma, \delta, q_0, F\rangle$. Una configuración instantánea es un par $(q, w) \in Q \times \Sigma *$ donde q es el estado en el que está el autómata y w es la cadena que resta consumir

Definición 4.1.2: Llamamos transición a la siguiente relación sobre $Q \times \Sigma^*$:

$$(q, w) \vdash (p, \beta) \text{ si } (\delta(q, a) = p \text{ y } w = a\beta)$$

De esto tenemos que $(q, \alpha\beta) \overset{*}{\vdash} (p, \beta) \iff \hat{\delta}(q, \alpha) = p$. Es decir, se puede llegar al estado p consumiendo la cadena α

Lema 4.1.1: Sea M = $\langle Q, \Sigma, \delta, q_0, F\rangle$ un AFD. Para todo $q \in Q$ y $\alpha, \beta \in \Sigma^*$ se tiene que:

$$\text{si } (q, \alpha\beta) \overset{*}{\vdash} (q, \beta) \text{ entonces } \forall i \geq 0, (q, \alpha^i\beta) \overset{*}{\vdash} (q, \beta)$$

Demostración: Por inducción sobre i:

- Caso base: $i = 0$
  $(q, \alpha^0\beta) \overset{0}{\vdash} (q, \beta)$
- Paso inductivo:
  Supongamos que vale para i, es decir si $(q, \alpha\beta) \overset{*}{\vdash} (q, \beta)$ entonces $(q, \alpha^i\beta) \overset{*}{\vdash} (q, \beta)$. Veamos que vale para i + 1:
  $(q, \alpha^{i+1}\beta) = (q, \alpha\alpha^i\beta) \underset{\substack{\text{por el antecedente}\\ \text{q asumimos}}}{\Longrightarrow} (q, \alpha\alpha^i\beta) \overset{*}{\vdash} (q, \alpha^i\beta) \underset{HI}{\overset{*}{\vdash}} (q, \beta)$

□

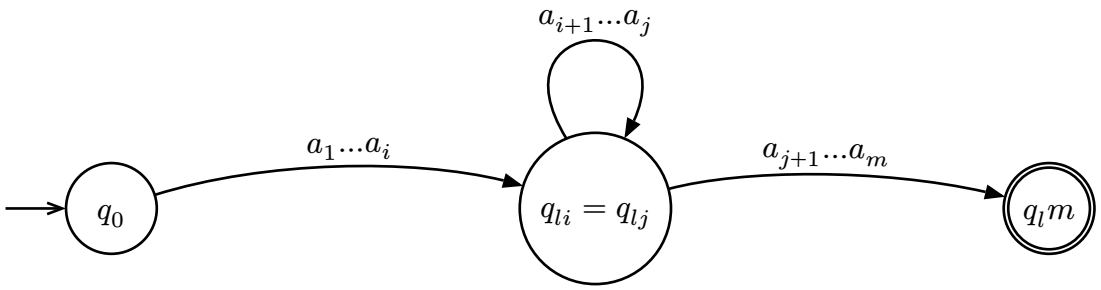
4.2. Lema de Pumping

Teorema 4.2.1: Sea L un lenguaje regular, entonces existe un número n tal que para toda cadena z en L con |z| ≥ n, existen cadenas u, v, w tales que:

- z = uvw,
- $|uv| \leq$ n,
- |v| ≥ 1

  $\forall$i ≥ 0, $uv^iw \in$ L

Demostración: Sea M un AFD tal que $\mathcal{L}(M)$ = L. Sea n su cantidad de estados. Sea z una cadena de longitud $m \geq n$, $z = a_1 ... a_m$ los símbolos que forman la cadena.
Para aceptar z usamos m transiciones, por lo tanto pasamos a través de m + 1 estados. Como $m + 1 > n$, tenemos que necesariamente debemos pasar al menos dos veces por un mismo estado para aceptar la cadena (pigeonhole principle).
Sea entonces $q_{l0} ... q_{lm}$ la sucesión de estados desde $q_0 (q_{l0})$ hasta un estado final (q_{lm})



Existen entonces j y k mínimos tales que $q_{lj} = q_{lk}$ con $0 \leq j < k \leq n$. Esto separa a z en tres subcadenas:

- $u = \begin{cases} a_1 ... a_j \text{ si } j>0 \\ \lambda \text{ si } j=0 \end{cases}$
- $v = a_{j+1} ... a_k$
- $w = \begin{cases} a_{k+1} ... a_m \text{ si } k<m \\ \lambda \text{ si } k=m \end{cases}$

Juntando todo esto, tenemos que:

$$|uv| \leq n$$
$$|v| \geq 0$$

y que:

$$(q_0, uvw) \overset{*}{\vdash} (q_{lj}, vw) \overset{*}{\vdash} (q_{lk}, w) \overset{*}{\vdash} (q_{lm}, \lambda)$$

Pero como $q_{lj} = q_{lk}$, y por el lema probado en la sección anterior, tenemos que:

$\forall i \geq 0 (q_{lj}, v^i w) \overset{*}{\vdash} (q_{lj}, w) = (q_{lk}, w)$ que tenemos alcanza un estado final

Por lo tanto, $uv^i w \in$ L, $\forall i \geq 0$ □

Ejemplo: Sea AFD $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, con $|Q|$ = n. Determinar veracidad y justificar:
1. $\mathcal{L}(M)$ es no vacío si y solo si existe $w \in \Sigma^*$ tal que $\hat{\delta}(q_0, w) \in F y\ |w| < n$

Demostración:

- $\Longleftarrow$) Es trivial ver que el lenguaje no es vacío
- $\Longrightarrow$) Supongamos que el lenguaje no es vacío y regular. Entonces, supongamos existe una cadena $z \in \mathcal{L}(M)$. Hay dos posibilidades, o bien la longitud de la cadena es menor a n, o bien es mayor. En el primer caso, no hace falta demostrar nada mas. En el segundo caso, por el lema de pumping, podemos descomponer la cadena en uvw tal que $|uv| \leq n$ y $|v| \geq 1$ y estar seguros que para todo i se cumple que $uv^i w \in \mathcal{L}(M)$. Luego por el lema de pumping, podemos "pumpear" a v con i = 0, reduciendo el tamaño de la cadena y repitiendo este proceso hasta llegar a una con longitud menor a n.

□

2. $\mathcal{L}(M)$ es infinito si y solo si existe $w \in \Sigma^*$ tal que $\hat{\delta}(q_0, w) \in F$ y $n \leq |w| < 2n$

Demostración:

- $\Longleftarrow$) Suponemos $z \in \mathcal{L}(M)$ y $n \leq |z| < 2n$. Por el lema de Pumping, tenemos $z = uvx$ con $|uv| \leq n$ y $|v| \geq 1$ y $uv^i x \in \mathcal{L}(M)$ para todo i. Luego $\mathcal{L}(M)$ es infinito
- $\Longrightarrow$) Supongamos $\mathcal{L}(M)$ es infinito y regular. Supongamos también que no existe una cadena z en el lenguaje con longitud entre n y 2n - 1. Primero notemos que como el lenguaje es infinito, necesariamente debe haber cadenas de longitud mayor a n (Pues lo único que puede aportar a la "infinitez"(? es que el tamaño de las cadenas pueda ser arbitrariamente grandes)

  Sin pérdida de generalidad, supongamos que |z| = 2n (Notar que si la longitud fuese mayor, Simplemente podríamos bombear hacia repetir el argumento hasta que se cumpla esta condición o se llegue al rango deseado. Notar también que no es posible " saltearse" el rango pues el mismo es de longitud n + 1, y saltearlo implicaría un ciclo que abarca más que la cantidad de estados del autómata)

  Por Lema de Pumping, tenemos que existen u, v, x tales que $z = uvx$, con $|uv\ | \leq n$, $v \geq 1$ y $\forall i uv^i x \in \mathcal{L}(M)$. En particular, $uv^0 x = ux$ está en el lenguaje.

  Como $|uvx| = 2n$ y $1 \leq |v| \leq n$ tenemos que $n \leq |ux| \leq 2n - 1$, contradiciendo nuestra suposición que dicha cadena no existía.

□

4.3. Propiedades de clausura de los lenguajes regulares

4.3.1. Unión

Teorema 4.3.1.1: El conjunto de lenguajes regulares incluidos en Σ^* es cerrado respecto de la unión

Demostración: Sean L_1 y L_2 lenguajes regulares. Sea $M_1 = \langle Q_1, \Sigma, \delta_1, q_1, F_1 \rangle$ tal que $\mathcal{L}(M_1) = L_1$ y $M_2 = \langle Q_2, \Sigma, \delta_2, q_2, F_2 \rangle$ tal que $\mathcal{L}(M_2) = L_2$ y $Q_1 \cap Q_2 = \emptyset$. definimos $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ tal que:

- $Q = Q_1 \times Q_2$
- $q_0 = (q_1, q_2)$
- $\delta((q, r), a) = (\delta_1(q, a), \delta_2(r, a))$ para $q \in Q_1, r \in Q_2, a \in \Sigma$
- $F = \{(p, q) : p \in F_1 \vee q \in F_2\}$

con $\mathcal{L}(M) = L_1 \cup L_2$.

Ahora queda demostrar la pertenencia

$$\begin{aligned} x \in \mathcal{L}(M) &\iff \delta\Big(\big(q_{1_0}, q_{2_0}\big), x\Big) \in F \\ &\iff \Big(\delta_1\big(q_{1_0}, x\big), \delta_2\big(q_{2_0}, x\big)\Big) \in F \\ &\iff \delta_1\big(q_{1_0}, x\big) \in F_1 \vee \delta_2\big(q_{2_0}, x\big) \in F_2 \\ &\iff x \in \mathcal{L}(M_1) \vee x \in \mathcal{L}(M_2). \end{aligned}$$

□

(Si no me equivoco, acá falta demostrar la equivalencia para $\hat{\delta}$, si no no se puede asumir nada sobre la aplicación de la misma a cadenas. No debería ser difícil, pero si llego debería corregir esto)

Una demostración alternativa es hacer uso de las expresiones regulares. Dados r_1 y r_2 expresiones regulares que denotan los lenguajes L_1 y L_2, respectivamente, podemos construir una expresión regular $r = r_1 \mid r_2$ que denota la unión de los lenguajes, por definición.

4.3.2. Concatenación y clausura de Kleene

Teorema 4.3.2.1: El conjunto de lenguajes regulares incluidos en Σ^* es cerrado respecto de la concatenación y la clausura de Kleene

(La demo más aceptada es hacer uso nuevamente de expresiones regulares, ya que por definición trivializan la prueba, pero me gustaría demostrar también que los autómatas que armamos son válidos, considerar agregarla / hacerla como ejercicio)

4.3.3. Complemento

Teorema 4.3.3.1: El conjunto de lenguajes regulares incluidos en Σ^* es cerrado respecto del complemento

Demostración: Sea L = $\mathcal{L}(M)$ para algún AFD M = $\langle Q, \Sigma, \delta, q_0, F \rangle$ cuya función de transición δ está completamente definida. Tenemos entonces que el autómata:

$$M' = \langle Q, \Sigma, \delta, q_0, Q - F \rangle$$

Como tenemos que ahora todas las cadenas que no eran aceptadas por M lo son, y viceversa, tenemos que $\mathcal{L}(M') = \Sigma^* - L = \overline{L}$. □

4.3.4. Intersección

Teorema 4.3.4.1: El conjunto de lenguajes regulares incluidos en Σ^* es cerrado respecto de la intersección

Demostración: Sea L_1 y L_2 lenguajes regulares. Sea $M_1 = \langle Q_1, \Sigma, \delta_1, q_{0_1}, F_1 \rangle$ tal que $\mathcal{L}(M_1) = L_1$ y $M_2 = \langle Q_2, \Sigma, \delta_2, q_{0_2}, F_2 \rangle$ tal que $\mathcal{L}(M_2) = L_2$. Definamos $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ tal que:

- $Q = Q_1 \times Q_2$
- $q_0 = \left(q_{0_1}, q_{0_2}\right)$
- $\delta((q, r), a) = (\delta_1(q, a), \delta_2(r, a))$ para $q \in Q_1, r \in Q_2, a \in \Sigma$
- F = $F_1 \times F_2 = \{(p, q) : p \in F_1 \wedge q \in F_2\}$ =

Con una inducción sobre $|w|$ es fácil ver que $\hat{\delta}((q, r), w) = \left(\hat{\delta}_1(q, w), \hat{\delta}_2(r, w)\right)$. Con esto en cuenta, tenemos que:

$$\begin{aligned} w \in \mathcal{L}(M) &\iff \hat{\delta}((p, r), w) \in F \\ &\iff \left(\hat{\delta}_1(p, w), \hat{\delta}_2(r, w) \in F_1 \times F_2\right)) \\ &\iff \left(\hat{\delta}_1(p, w) \in F_1\right) \wedge \left(\hat{\delta}_2(r, w) \in F_2\right) \\ &\iff w \in L_1 \wedge w \in L_2 \end{aligned}$$

□

4.4. Unión e Intersección finitos de lenguajes regulares

Teorema 4.4.1: $\forall n \in \mathbb{N}, \bigcup_{i=1}^{n} L_i$ es regular

Teorema 4.4.2: $\forall n \in \mathbb{N}, \bigcap_{i=1}^{n} L_i$ es regular

Demostración: Demostramos la unión por inducción sobre n, la intersección es similar:

- Caso base: $n = 0$
  $\bigcup_{i=1}^{0} L_i = \emptyset$, que sabemos es regular
- Paso inductivo:
  Supongamos que $\bigcup_{i=1}^{n} L_i$ es regular. Sea $L_{\{n+1\}}$ un lenguaje regular.

  Entonces, por la propiedad de clausura de la unión, tenemos que $\bigcup_{i=1}^{n+1} L_i = \bigcup_{i=1}^{n} L_i \cup L_{\{n+1\}}$ es regular.

□

Lema 4.4.3: Los lenguajes regulares no están clausurados por union infinita

Demostración: Sea $L_i = \{a^i b^i\}$, si lo estuvieran, entonces $\bigcup_{i=1}^{\infty} L_i$ sería regular, pero:

$\bigcup_{i=1}^{\infty} L_i = \bigcup_{i=1}^{\infty} \{a^i b^i\} = \{a^k b^k : k \in \mathbb{N}\}$, que sabemos no es regular

□

4.5. Todo lenguaje finito es regular

Teorema 4.5.1: Todo lenguaje finito es regular

Demostración: Sea L un lenguaje finito, con n cadenas $\{w_1, w_2, ..., w_n\}$. Para cada $1 \leq i \leq n$, sea $L_i = \{w_i\}$

Entonces $L = \bigcup_{i=1}^{n} L_i$, que sabemos es regular pues cada L_i lo es □

4.6. Problemas de decisibilidad de lenguajes regulares

4.6.1. Vacuidad

Teorema 4.6.1.1: El problema de la vacuidad de un lenguaje regular es decidible

Demostración: suponiendo que la representación del lenguaje está dada mediante un AFD (notar que siempre se puede convertir de una representación a otra, y en esta materia estamos ignorando complejidad, porque el plan nuevo de la carrera es **muy bueno**)

- Se determina el conjunto de estados alcanzables desde el estado inicial
- Si ninguno de los estados alcanzables es final, entonces el lenguaje es vacío

□

4.6.2. Pertenencia

Teorema 4.6.2.1: El problema de la pertenencia de un lenguaje regular es decidible

Demostración: Suponiendo que la representación del lenguaje está dada mediante un AFD, se puede determinar si una cadena pertenece simplemente determinando si la misma es aceptada por el autómata □

4.6.3. Finititud

Teorema 4.6.3.1: El problema de la finitud de un lenguaje regular es decidible

Demostración: Anteriormente demostramos cual era la condición suficiente y necesaria para que un lenguaje fuese infinito □

4.6.4. Equivalencia

Teorema 4.6.4.1: El problema de la equivalencia de dos lenguajes regulares es decidible

Demostración: $L_1 \Delta L_2 = \left(L_1 \cap \overline{L_2}\right) \cup \left(\overline{L_1} \cap L_2\right) = \emptyset \iff L_1 = L_2$ □

(**El libro propone una demostración más constructiva y eficiente, introduciendo el concepto de equivalencia de estados**)

5. Gramáticas Libres de Contexto

Definición 5.1: Recordemos que, llamamos a una gramática $G = \langle V_N, V_T, P, S \rangle$ libre de contexto si las producciones P son de la forma:

$$A \to \alpha \text{ con } A \in V_N \text{ y } \alpha \in (V_N \cup V_T) *$$

5.1. Derivación

Si $\alpha, \beta, \gamma_1, \gamma_2 \in (V_N \cup V_T)^*$ y $\alpha \to \beta \in P$ entonces:

$$\gamma_1 \alpha \gamma_2 \Rightarrow \gamma_1 \beta \gamma_2$$

La relación $\Rightarrow$ es un subconjunto de $(V_N \cup V_T)^* \times (V_N \cup V_T)^*$ y significa derivar en un solo paso

Las relaciones $\overset{*}{\Rightarrow}$ y $\overset{+}{\Rightarrow}$ son las clausuras reflexiva y transitiva de $\Rightarrow$, respectivamente

Si $\alpha \in (V_N \cup V_T)^* y \ \ S \overset{*}{\Rightarrow} \alpha$ decimos que la misma es una forma sentencial de G.

5.2. Inferencia

Definición 5.2.1: Una manera alternativa de determinar la pertenencia de una cadena es partir desde el cuerpo de las producciones, identificando las cadenas que ya sabemos pertenecen al lenguaje , y concatenandolas formando así cadenas más complejas

5.3. Lenguaje de una Gramática

Definición 5.3.1: El lenguaje de una gramática G, denotado $\mathcal{L}(G)$ es el conjunto de todas las cadenas que pueden ser derivadas desde el símbolo inicial S de G. Es decir:

$$\mathcal{L}(G) = \left\{ w \in V_T^* : S \overset{+}{\Rightarrow} w \right\}$$

5.4. árbol de derivación

Definición 5.4.1: Un árbol de derivación para una gramática G es un árbol tal que:

- Sus hojas están etiquetadas con símbolos de V_T, V_N o λ, caso en el cual el mismo debe ser el único hijo de su padre.
- Sus nodos internos están etiquetados con símbolos de V_N.
- La raíz está etiquetada con el símbolo inicial S
- Si un nodo interno está etiquetado con A y sus hijos están etiquetados con $X_1, X_2, ..., X_n$ entonces $A \to X_1 X_2 ... X_n$ es una producción de G

El árbol de derivación correspondiente a una variable A lo notaremos con $\mathcal{T}(A)$

Definición 5.4.2: Llamamos producto (yield) de un árbol de derivación a la cadena que se obtiene al concatenar las etiquetas de las hojas de izquierda a derecha. En particular, tenemos que aquellos árboles que cumplen que:

- Todas las hojas están etiquetadas con símbolos de V_T o con λ
- La raíz está etiquetada con el símbolo inicial S

Son los árboles cuyo producto son las cadenas en el lenguaje de la gramática correspondiente

Definición 5.4.3: LLamamos camino de X en un árbol $\mathcal{T}(A)$ a la cadena $A, X_1, ..., X_k, X$ tal que:

$$A \Rightarrow .....X_1..... \Rightarrow .....X_2..... \Rightarrow .... \Rightarrow .....X_k..... \Rightarrow X$$

Definición 5.4.4: Llamamos altura de $\mathcal{T}(A)$ a:

$$\{\max\{|ax| : x \text{ es una hoja de } \mathcal{T}(A) \text{ y } Aax \text{ es un camino de x}\}\}$$

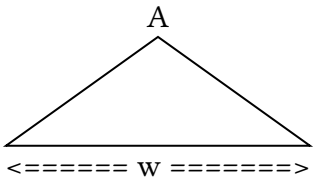
5.5. Equivalencia inferencia, árbol Derivación y Derivaciones

Teorema 5.5.1: Sea G una GLC. Si el procedimiento de inferencia recursivo determina que una cadena w con sólo terminales esta en el lenguaje de una gramática definido por una variable, entoces hay una árbol de derivación con raíz en esa variable que produce w

Demostración: Por inducción en la cantidad de pasos utilizados para inferir que w esta en el lenguaje de la variable A:

- Caso base: Un sólo paso.

En este caso, tenemos que $A \to w$ debe ser una producción de la gramática, y por lo tanto, el árbol de derivación es simplemente un nodo con raíz en A y hojas etiquetadas con los símbolos de w.



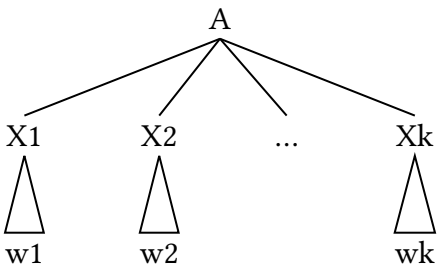
- Paso inductivo:

Supongamos que el hecho de que w esté en el lenguaje es inferido luego de n + 1 pasos, y que el teorema vale para todas las cadenas x y variables B tal que su pertenencia fue determinada con n o menos pasos. Consideremos el último paso de la inferencia utilizada para determinar que w está en el lenguaje de A. Por definición, la inferencia usa alguna producción de A, supongamos que es $A \to X_1X_2...X_k$, donde cada X_i es o bien una variable o un terminal. Podemos entonces tomar $w = w_1...w_k$como:

- Si X_i es un terminal, entonces $X_i = w_i$, es decir, w_i solo consiste de este terminal en la producción

- Si X_i es una variable, entonces w_i es una cadena que fue inferida con n pasos que está en el lenguaje de esa variable, y por hipótesis inductiva, sabemos que hay un árbol de derivación con raíz en X_i que produce w_i

  Con esto, podemos armar el siguiente árbol de derivación, notar que en el primer caso los subárboles correspondiientes son triviales de un solo nodo:



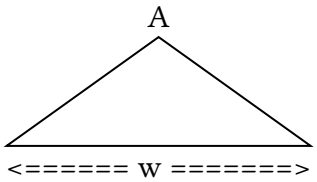
□

Teorema 5.5.2: Sea G una GLC. Si hay un árbol de derivación con raíz en una variable A que produce una cadena w, tal que $w \in V_T^*$. Entonces hay una derivación a izquierda $A \underset{L}{\overset{*}{\Rightarrow}} w$ en la gramática

Demostración: Por inducción en la altura del árbol de derivación:

- Caso base: Altura 1

  En este caso, tenemos que el árbol de derivación es simplemente un nodo con raíz en A y hojas etiquetadas con los símbolos de w. Como se trata de un arbol de derivación, $A \to w$ debe ser una producción, por lo que la derivación es simplemente $A \underset{L}{\Rightarrow} w$



- Paso inductivo:

  Supongamos que el árbol de derivación tiene altura n + 1. Entonces, la raíz del árbol de derivación es A, y tiene hijos $X_1, X_2, ..., X_k$:

  - Si X_i es un terminal, definimos w_i como la cadena que consiste solo de X_i, es decir $w_i = X_i$
  - Si X_i es una variable, entonces debe ser la raíz para algún subárbol con una producción de sólo terminales (pues si no no se cumpliría el precedente), a la que identificamos como w_i. Notar que por hipótesis inductiva, sabemos que hay una derivación a izquierda $X_i \underset{L}{\overset{*}{\Rightarrow}} w_i$.

  Ahora podemos armar la derivación a izquierda $A \underset{L}{\overset{*}{\Rightarrow}} w$ de la siguiente manera:

$$A \underset{L}{\Rightarrow} X_1X_2...X_k \underset{L}{\overset{*}{\Rightarrow}} w_1X_2...X_k \underset{L}{\overset{*}{\Rightarrow}} w_1w_2...X_k \underset{L}{\overset{*}{\Rightarrow}} w_1w_2...w_k = w$$

□

Teorema 5.5.3: Sea G una GLC. Si hay una derivación $A \overset{*}{\Rightarrow} w$ en la gramática, entonces el procedimiento de inferencia recursivo determina que w está en el lenguaje de la variable A

Demostración: Por inducción en la cantidad de pasos de la derivación:

- Caso base: Un solo paso

  En este caso, tenemos que $A \to w$ debe ser una producción de la gramática, y por lo tanto, el procedimiento de inferencia recursivo determina que w está en el lenguaje de A de manera inmediata.

- Paso inductivo:

  Supongamos que la derivación tiene n + 1 pasos, y que la hipótesis vale para cualquier derivación con menos de n pasos. Escribiendo la primera derivación de la forma $A \Rightarrow X_1X_2...X_k$, separamos a $w = w_1...w_k$ tal que:
  - Si X_i es un terminal, entonces $X_i = w_i$, es decir, w_i solo consiste de este terminal en la producción
  - Si X_i es una variable, entonces $X_i \Rightarrow w_i$. Como sabemos que $A \overset{*}{\Rightarrow} w$ no es parte de esta derivación, (pues la misma tiene más pasos) tenemos que la misma tiene menos de n pasos, por lo que, por H.I sabemos que se puede inferir que w_i está en X_i

  Por lo tanto, tenemos una producción de la forma $A \to X_1X_2...X_k$, con cada w_i quivalente a X_i o en el lenguaje del mismo, por lo tanto, el procedimiento de inferencia recursivo determina en el siguiente paso que w está en el lenguaje de A

□

5.6. Gramáticas Ambiguas

Definición 5.6.1: Una gramática es ambigua si existe al menos una cadena $w \in V_T*$ para la cual haya al menos dos árboles de derivación, ambos con raíz S y que produzcan w

Teorema 5.6.1: Para toda gramática G y cadena $w \in T^*$, w tiene dos árboles de derivación distintos $\iff$ w tiene dos derivaciones a izquierda distintas a partir de S

Demostración: □

Definición 5.6.2: Un Lenguaje libre de contexto es inherentemente ambiguo si no existe una gramática no ambigua que genere el lenguaje

6. Autómatas de Pila

Definición 6.1: Un autómata de pila es una tupla $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ donde:
- Q es un conjunto finito de estados
- Σ es un alfabeto finito de entrada
- Γ es un alfabeto finito de la pila
- $q_0 \in Q$ es el estado inicial
- $Z_0 \in \Gamma$ es la configuración inicial de la pila
- $F \subseteq Q$ es el conjunto de estados finales
- $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to P(Q \times \Gamma^*)$ es la función de transición

La interpretación de $\delta(q, x, Z) = \{(p_1, \gamma_1), ..., (p_n, \gamma_n)\}$ es como sigue:

Cuando el estado del autómata es q, el símbolo que la cabeza lectora está inspeccionando en ese momento es x, y en el tope de la pila hay Z, se realizan las sigueintes acciones:

- Si $x \in \Sigma$ ($\neq \lambda$), la cabeza lectora avanza una posición para inspeccionar el siguiente símbolo
- Se elimina el símbolo Z del tope de la pila
- Se selecciona un par (p_i, γ_i) entre los existentes
- Se apila la cadena $\gamma_i = c_1 c_2 ... c_k$, con $c_i \in \Gamma$ en la pila del autómata, con el símbolo c_1 quedando en el tope
- Se cambia el control del autómata al estado p_i

6.1. Configuración de un AP

Definición 6.1.1: Una configuración instánea de un AP es una terna (q, w, γ) donde:

- q es el estado actual del autómata
- w es la cadena de entrada que queda por leer
- γ es el contenido de la pila

La configuración inicial de un AP para una cadena w_0 es (q_0, w_0, Z_0)

Definición 6.1.2: Representamos al cambio entre configuraciones instantáneas de un autómata tal que, para todo $x \in \Sigma, w \in \Sigma^*, Z \in \Gamma, \gamma, \pi \in \Gamma^*, q, p \in Q$:

- $(q, xw, Z\pi) \vdash (p, w, \gamma\pi)$ si $(p, \gamma) \in \delta(q, x, Z)$
- $(q, w, Z\pi) \vdash (p, w, \gamma\pi)$ si $(p, \gamma) \in \delta(q, \lambda, Z)$

Teorema 6.1.1: Información que el AP no utiliza no afecta su comportamient. Formalmente, sea $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ un AP, y $(q, x, \alpha) \overset{*}{\vdash} (p, v, \beta)$, entonces, para toda cadena $w \in \Sigma^*$ y $\gamma \in \Gamma^*$:

$$(q, xw, \alpha\gamma) \overset{*}{\vdash} (p, vw, \beta\gamma)$$

Demostración: Se trata de una simple inducción sobre la cantidad de pasos en los cambios de configuración □

6.2. Lenguaje de un AP

Definición 6.2.1: El lenguaje de un AP P, denotado $\mathcal{L}(P)$ es el conjunto de todas las cadenas que consume y a la vez alcanzan un estado final, es decir:

$$\mathcal{L}(P) = \left\{ w \in \Sigma^* : \exists (p \in F, \gamma \in \Gamma^*) \ \ (q_0, w, Z_0) \overset{*}{\vdash} (p, \lambda, \gamma) \right\}$$

Definición 6.2.2: El lenguaje reconocido por P por pila vacía es:

$$\mathcal{L}_{\lambda(P)} = \left\{ w \in \Sigma^* : \exists (p \in Q) \ \ (q_0, w, Z_0) \overset{*}{\vdash} (p, \lambda, \lambda) \right\}$$

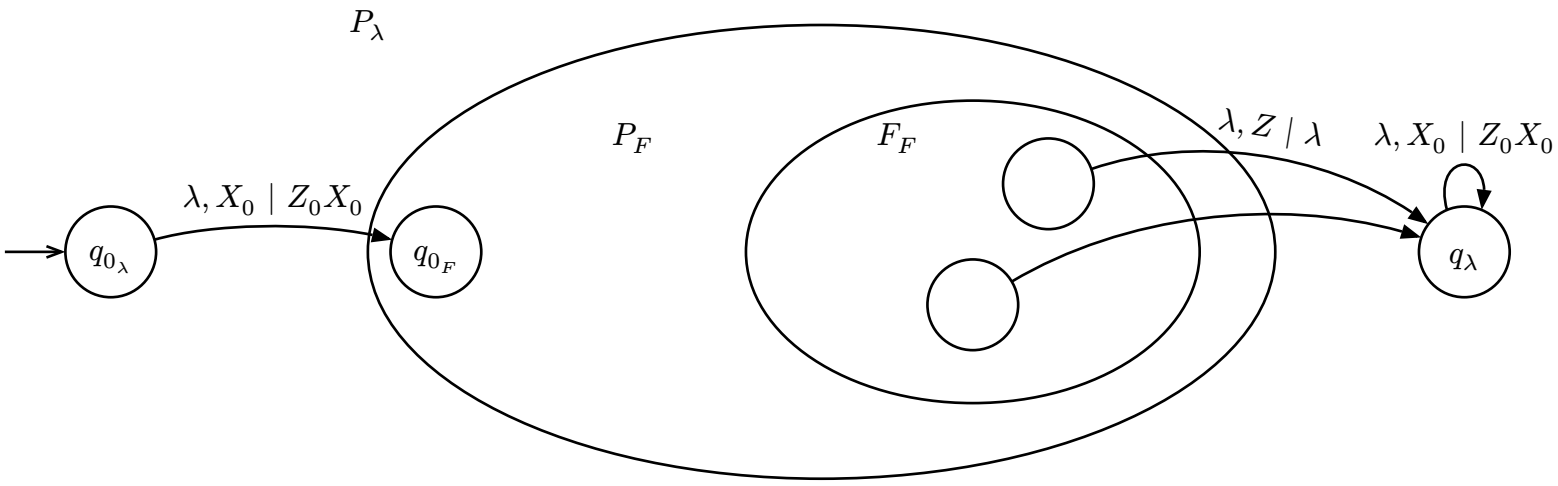
6.3. Equivalencia $\mathcal{L}(P)$ y $\mathcal{L}_\lambda(P)$

Teorema 6.3.1: Para todo AP $P_F = \langle Q_F, \Sigma, \Gamma_F, \delta_F, q_{0_F}, Z_0, F_F \rangle$, existe un AP P_λ tal que:

$$\mathcal{L}(P_F) = \mathcal{L}_\lambda(P_\lambda)$$

Demostración: Definimos $P_\lambda = \langle Q_F \cup \left\{ q_{0_\lambda}, q_\lambda \right\}, \Sigma, \Gamma \cup \{X_0\}, \delta_\lambda, q_{0_\lambda}, \emptyset \rangle$ tal que:

1. $\delta_\lambda\left(q_{0_\lambda}, w, X_0\right) = \left\{ \left(q_{0_F}, Z_0 X_0\right) \right\}$
2. $\forall (q \in Q_F, x \in \Sigma \cup \{\lambda\}, Z \in \Gamma_F), \delta_\lambda(q, x, Z) = \delta_F(q, x, Z)$
3. $\forall (q \in F_F, Z \in \Gamma_F \cup \{X_0\}), (q_\lambda, \lambda) \in \delta_\lambda(q, \lambda, Z)$
4. $\forall (Z \in \Gamma_F \cup \{X_0\}), (q_\lambda, \lambda) \in \delta_\lambda(q_\lambda, \lambda, Z)$



Queremos ahora ver que $w \in \mathcal{L}(P_F) \iff w \in \mathcal{L}_\lambda(P_\lambda)$:

- $\Longrightarrow$) Si $w \in \mathcal{L}(P_F)$ entonces $\left(q_{0_F}, w, Z_0\right) \overset{*}{\underset{P_F}{\vdash}} (q, \lambda, \gamma)$, con $q \in F_F, \gamma \in \Gamma^*$

Por la regla 1 de δ_λ, $\delta_\lambda\left(q_{0_\lambda}, \lambda, X_0\right) = \left\{ \left(q_{0_f}, Z_0 X_0\right) \right\}$, por lo que:

$$q_{0_\lambda}, w, X_0 \underset{P_\lambda}{\vdash} \left(q_{0_F}, w, Z_0 X_0\right)$$

Por la regla 2 de δ_λ, tenemos que todas las transiciones de P_F están en P_λ (lo "simula"), por lo que:

$$\left(q_{0_F}, w, Z_0\right) \overset{*}{\underset{P_\lambda}{\vdash}} (q, \lambda, \gamma)$$

Entonces:

$$\left(q_{0_\lambda}, w, X_0\right) \underset{P_\lambda}{\vdash} \left(q_{0_F}, w, Z_0 X_0\right) \overset{*}{\underset{P_\lambda}{\vdash}} (q, \lambda, \gamma X_0)$$

Nuevamente por las regals 3 y 4 δ_λ, para todo estado final q $\in F_F$ y $Z \in \Gamma \cup \{X_0\}$ tenemos que una vez que se alcanza un estado final en P_F, se puede llegar a P_λ con la pila vacía, por lo que:

$$(q, \lambda, \gamma X_0) \underset{P_\lambda}{\vdash} (q_\lambda, \lambda, \gamma X_0)$$

Y que, una vez en q_λ, se puede llegar a la pila vacía sin importar el tope de la pila, por lo que:

$$(q_\lambda, \lambda, \gamma) \overset{*}{\underset{P_\lambda}{\vdash}} (q_\lambda, \lambda, \lambda)$$

Juntando todo, tenemos:

$$\left(q_{0_\lambda}, w, X_0\right) \underset{P_\lambda}{\vdash} \left(q_{0_F}, w, Z_0 X_0\right) \overset{*}{\underset{P_\lambda}{\vdash}} (q, \lambda, \gamma X_0) \underset{P_\lambda}{\vdash} (q_\lambda, \lambda, \gamma X_0) \overset{*}{\underset{P_\lambda}{\vdash}} (q_\lambda, \lambda, \lambda)$$

Por lo que si $w \in \mathcal{L}(P_F)$, entonces $w \in \mathcal{L}_\lambda(P_\lambda)$

- $\Longleftarrow$) Tenemos que $w \in \mathcal{L}_\lambda(P_\lambda)$. Sabemos por la definición de δ_λ que la única manera de que P_λ vacíe su pila es mediante el estado p_λ (pues es el único estado que puede identificar X_0), además, sabemos que la única manera en la que P_λ alcance este estado, es si el autómata simulado alcanza un estado final de P_F. Finalmente, sabemos que el primer movimiento del autómata es siempre saltar al autómata simulado, por lo que tenemos:

$$\left(q_{0_\lambda}, w, X_0\right) \underset{P_\lambda}{\vdash} \underbrace{\left(q_{0_F}, w, Z_0 X_0\right) \overset{*}{\underset{P_\lambda}{\vdash}} (q, \lambda, \gamma X_0)}_{A} \underset{P_\lambda}{\vdash} (q_\lambda, \lambda, \gamma X_0) \overset{*}{\underset{P_\lambda}{\vdash}} (q_\lambda, \lambda, \lambda)$$

Pero la tranisción en A necesariamente implica que (pues la misma no hace uso de X_0):

$$\left(q_{0_F}, w, Z_0\right) \overset{*}{\underset{P_F}{\vdash}} (q, \lambda, \gamma)$$

Por lo que si $w \in \mathcal{L}_\lambda(P_\lambda)$, entonces $w \in \mathcal{L}(P_F)$ □

Teorema 6.3.2: Para todo AP $P_\lambda = \langle Q_\lambda, \Sigma, \Gamma_\lambda, \delta_\lambda, q_{0_\lambda}, X_0, \emptyset \rangle$, existe un AP P_F tal que:

$$\mathcal{L}_\lambda(P_\lambda) = \mathcal{L}(P_F)$$

Demostración: Definimos $P_F = \langle Q_\lambda \cup \left\{q_{0_F}, q_f\right\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, q_{0_\lambda}, F_\lambda \rangle$ tal que:

1. $\delta\left(q_{0_F}, \lambda, Z_0\right) = \left\{\left(q_{0_\lambda}, X_0 Z_0\right)\right\}$
2. $\forall (q \in Q_\lambda, x \in \Sigma \cup \{\lambda\}, Z \in \Gamma_\lambda), \delta_{F(q,x,Z)} = \delta_\lambda(q, x, Z)$
3. $\forall q \in Q_\lambda, \left(q_f, \lambda\right) \in \delta_{F(q,\lambda,Z_0)}$

Ahora tenemos que demostrar que $w \in \mathcal{L}(P_F) \iff w \in \mathcal{L}_\lambda(P_\lambda)$:

- $\Longleftarrow$) Si $w \in \mathcal{L}_\lambda(P_\lambda)$ entonces tenemos que $\left(q_{0_\lambda}, w, X_0\right) \overset{*}{\underset{P_\lambda}{\vdash}} (q, \lambda, \lambda)$

Por definición de delta, tenemos que

$$\left(q_{0_F}, w, Z_0\right) \underset{P_F}{\vdash} \left(q_{0_\lambda}, w, X_0 Z_0\right) \overset{*}{\underset{P_F}{\vdash}} (q, \lambda, \lambda) \underset{P_F}{\vdash} (q_F, \lambda, \lambda)$$

Y por lo tanto $w \in \mathcal{L}(P_F)$ □

- $\Longrightarrow$) Si $w \in \mathcal{L}(P_F)$, tenemos que:

$$\left(q_{0_F}, w, Z_0\right) \underset{P_F}{\vdash} \left(q_{0_\lambda}, w, X_0 Z_0\right) \overset{*}{\underset{P_F}{\vdash}} (p, \lambda, Z_0) \underset{P_F}{\vdash} \left(q_f, \lambda, \lambda\right)$$

Pero por definición de P_F:

$$\left(q_{0_\lambda}, w, X_0 Z_0\right) \underset{P_F}{\vdash} (p, \lambda, Z_0) \iff \left(q_{0_\lambda}, w, X_0\right) \underset{P_\lambda}{\vdash} (p, \lambda, \lambda)$$

Luego, tenemos que $\left(q_{0_\lambda}, w, X_0\right) \underset{P_\lambda}{\vdash} (p, \lambda, \lambda)$, por lo que $w \in \mathcal{L}_\lambda(P_\lambda)$

6.4. Equivalencia GLCs y APDs

Teorema 6.4.1: Para toda GLC G, existe un AP M tal que:

$$\mathcal{L}(G) = \mathcal{L}_\lambda(M)$$

Demostración: Sea G = $\langle V_N, V_T, P, S\rangle$ una GLC , definimos M = $\langle \{q\}, V_T, V_T \cup V_N, \delta, q, S, \emptyset\rangle$ tal que:

- $\delta : Q \times (V_T \cup \lambda) \times (V_N \cup V_T) \to \mathcal{P}\left(Q \times (V_T \cup V_N)^*\right)$
  - Si $A \to \alpha \in P$, entonces $\delta(q, \lambda, A) \ni \{(q, \alpha)\}$ (Para toda producción en P, M lo simula desapilando la variable correspondiente y pusheando el cuerpo de la producción)
  - Para todo $x \in V_T$, $\delta(q, x, x) = \{(q, \lambda)\}$ (Si se tiene que el tope de la pila es un terminal, se desapila y se avanza si es igualal símbolo que la cabeza lectora está leyendo)

Queremoe ver que $w \in \mathcal{L}(G) \iff w \in \mathcal{L}_\lambda(M)$:

Lema 6.4.2: $\forall (A \in V_N, w \in V_T\, *) A \overset{+}{\Rightarrow} w$ sii $(q, w, A) \overset{*}{\underset{M}{\vdash}} (q, \lambda, \lambda)$

Primero demostramos este lema por inducción sobre m, la cantidad de pasos en la derivación:

- Caso base: m = 1

  En este caso, tenemos que $A \overset{1}{\Rightarrow} w$ para $w = x_1 x_2 ... x_k$. Notar que si esta es una derivación posible para w, entonces necesariamente tiene que haber una producción $A \to x_1 x_2 ... x_k \in P$, luego, por la primer regla de δ, seguida por la aplicación de la segunda (que nos permite eliminar terminales del tope siempre que matcheen con la cadena), tenemos:

$$(q, w, A) \underset{M}{\vdash} (q, x_1 x_2 ... x_k, x_1 x_2 ... x_k) \overset{k}{\underset{M}{\vdash}} (q, \lambda, \lambda)$$

- Paso inductivo:

  Por H.I, tenemos que, para todo $j < m$, $A \overset{j}{\Rightarrow} w$ sii $(q, w, A) \overset{j}{\underset{M}{\vdash}} (q, \lambda, \lambda)$

Sea $w = w_1...w_k$, por definición de derivación, tenemos que $A \overset{m}{\Rightarrow} w$ sii $A \to X_1X_2...X_k$ es una producción de la gramática, y que $X_1 \overset{m_1}{\Rightarrow} w_1, X_2 \overset{m_2}{\Rightarrow} w_2, ..., X_k \overset{m_k}{\Rightarrow} w_k$, para $m_i < m$.

Por otro lado, por def. de M tenemos que $A \to X_1...X_k \in P$ sii $(q, w, A) \underset{M}{\vdash} (q, w, X_1...X_k)$

Si $X_i \in V_N$, entonces por H.I $(q, w_i, X_i) \overset{*}{\underset{M}{\vdash}} (q, \lambda, \lambda)$

Si $X_i \in V_T, X_i = w_i$ y entonces por la segunda regla de $\delta, (q, w_i, X_i) \underset{M}{\vdash} (q, \lambda, \lambda)$

Juntando todo:

$$(q, w, A) \underset{M}{\vdash} (q, w_1...w_k, X_1...X_k) \overset{*}{\underset{M}{\vdash}} (q, \lambda, \lambda)$$

Volviendo al teorema, el lema nos dice que $A \overset{+}{\Rightarrow} w$ sii $(q, w, A) \overset{*}{\underset{M}{\vdash}} (q, \lambda, \lambda)$, luego, tomando S = A obtenemos la prueba del teorema

□

Teorema 6.4.3: Si M es un APD, entonces existe una GLC G tal que:

$$\mathcal{L}_\lambda(M) = \mathcal{L}(G)$$

Demostración: Sea M = $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset \rangle$ un APD, definimos G = $\langle V_N, V_T, P, S \rangle$ tal que:

- $V_N = \{[qZp] : q, p \in Q, Z \in \Gamma\} \cup \{S\}, V_T = \Sigma$, y P dado por:
  - $S \to [q_0 Z_0 q]$, para cada p en Q
  - $[qZq_1] \to x$ sii $(q_1, \lambda) \in \delta(q, x, Z)$
  - $[qZq_1] \to \lambda$ sii $(q_1, \lambda) \in \delta(q, \lambda, Z)$
  - Para cada $q, q_1, ...q_m + 1 \in Q, x \in \Sigma$ y $Z, Y_1...Y_m \in \Gamma$:
    - $[qZq_{m+1}] \to x[q_1Y_1q_2]...[q_mY_mq_{m+1}]$ en P sii $(q_1, Y_1...Y_m) \in \delta(q, x, Z)$
    - $[qZq_{m+1}] \to [q_1Y_1q_2]...[q_mY_mq_{m+1}]$ en P sii $(q_1, Y_1...Y_m) \in \delta(q, \lambda, Z)$

Antes de coninuar, primero probamos el siguiente lema:

Lema 6.4.4: Para todo $q, p \in \Sigma, Z \in \Gamma$:

$$(q, w, Z) \overset{*}{\underset{M}{\vdash}} (p, \lambda, \lambda) \text{ sii } [qZp] \overset{*}{\underset{G}{\Rightarrow}} w$$

- $\Longrightarrow$) Veamos por inducción sobre i que vale:

$$\text{Si } (q, w, Z) \overset{i}{\underset{M}{\vdash}} (p, \lambda, \lambda) \text{ entonces } [qZp] \overset{*}{\underset{G}{\Rightarrow}} w$$

  - Caso base: i = 1

    En este caso, tenemos que $(q, x, Z) \overset{1}{\underset{M}{\vdash}} (q, \lambda, \lambda)$, por lo que $(q, \lambda) \in \delta(q, x, Z)$, entonces,

por definición de G, tenemos que, $[qZp] \to x$. Por lo tanto, $[qZq] \underset{G}{\Rightarrow} x$

- Paso inductivo:

  Sea $w = x\alpha$, con $\alpha \in \Sigma*$, tenemos que $(q, w, Z) \overset{i}{\underset{M}{\vdash}} (p, \lambda, \lambda)$. Sean $Y_1...Y_k \in \Gamma$, tenemos que necesariamente el primer movimiento del apd debe ser :

  $$(q, x\alpha, Z) \underset{M}{\vdash} (q_1, \alpha, Y_1...Y_k) \overset{i-1}{\underset{M}{\vdash}} (p, \lambda, \lambda)$$

  Descomponiendo a $\alpha = \alpha_1...\alpha_k$ tales que α_j es el input consumido al terminar de popear a Y_j de la pila, y sean p_j y p_{j+1} los estados cuando se popea Y_j, y cuando se termina se tiene a Y_{j+1} al tope, respectivamente. Tenemos que:

  $$\left(q_i, \alpha_j, Y_j\right) \overset{k_i}{\underset{M}{\vdash}} \left(q_{j+1}, \lambda, \lambda\right)$$

  Pero como $k_i < i$, tenemos:

  $$\text{Si } \left(q_i, \alpha_j, Y_j\right) \overset{k_i}{\underset{M}{\vdash}} \left(q_{j+1}, \lambda, \lambda\right) \text{ entonces } \left[q_j Y_j q_{j+1}\right] \overset{*}{\underset{G}{\Rightarrow}} \alpha_j$$

  Como en G se tiene la producción $[qZp] \to x[q_1Y_1q_2]...[q_kY_kp]$, tenemos que podemos armar la derivación:

  $$[qZp] \Rightarrow x[q_1Y_1q_2][q_2Y_2q_3]...[q_kY_kp] \overset{*}{\Rightarrow} x\alpha_1[q_2Y_2q_3]...[q_kY_kp] \overset{*}{\Rightarrow} x\alpha_1\alpha_2...\alpha_k = w$$

- $\Longleftarrow$) Veamos por inducción sobre i que vale:

  $$\text{Si } [qZp] \overset{i}{\Rightarrow} w \text{ entonces } (q, w, Z) \overset{*}{\underset{M}{\vdash}} (p, \lambda, \lambda)$$

- Caso base: i = 1

  En este caso, tenemos que $[qZp] \overset{1}{\Rightarrow} x$, por lo que necesariamente $[qZp] \to x$ es una producción de G y por definición$(q, x, Z) \underset{M}{\vdash} (p, \lambda, \lambda)$, por lo que $(q, x, Z) \in \delta(q, x, Z)$

- Paso inductivo:

  Tenemos que $[qZp] \overset{i}{\underset{G}{\Rightarrow}} w$, pero sabemos que necesariamente el primer paso de la derivación debe ser y ser continuada por una derivación del estilo:

  $$[qZp] \Rightarrow x[q_1Y_1q_2]...[q_nY_np] \overset{i-1}{\Rightarrow} w \tag{3}$$

  Sea entonces $w = xw_1...w_n$ de manera tal que, para $1 \leq j \leq n, k_i < i$ se cumpla que:

  $$\left[q_j Y_j q_{j+1}\right] \overset{k_i}{\Rightarrow} w_j \text{ (Es decir, la variable deriva en } w_j)$$

  Por H.I, tenemos que se cumple que, $\left(q_j, w_j, Y_j\right) \overset{*}{\underset{M}{\vdash}} \left(q_{j+1}, \lambda, \lambda\right)$

  Notemos que también se cumple que $\left(q_j, w_j, Y_jY_{j+1}...Y_n\right) \overset{*}{\underset{M}{\vdash}} \left(q_{j+1}, \lambda, Y_{j+1}...Y_n\right)$

  Por lo que tenemos por como construimos G que hay un cambio de configuración:

  $$(q, x, Z) \underset{M}{\vdash} (q_1, \lambda, Y_1...Y_n)$$

  Juntando todo:

  $$(q, xw_1...w_n, Z) \underset{M}{\vdash} (q_1, w_1...w_n, Y_1...Y_n) \overset{*}{\underset{M}{\vdash}} (p, \lambda, \lambda)$$

Con el lema probado, podemos tomar q = q_0 y Z = Z_0 tal que nos queda:

$$(q_0, w, Z_0) \overset{*}{\underset{M}{\vdash}} (p, \lambda, \lambda) \text{ sii } [q_0 Z_0 p] \overset{*}{\underset{G}{\Rightarrow}} w$$

Por la definición de G, $S \to [q_0 Z_0 p] \in P$, luego:

$$(q_0, w, Z_0) \overset{*}{\underset{M}{\vdash}} (p, \lambda, \lambda) \text{ sii } S \overset{*}{\underset{G}{\Rightarrow}} w$$

Que es equivalente a decir que:

$$w \in \mathcal{L}_\lambda(M) \text{ sii } w \in \mathcal{L}(G)$$

□

6.5. Autómatas de pila deterministicos

Definición 6.5.1: Un autómata de pila es determinístico si para todo estado q, símbolo x y Z en la pila se cumple que:

- $\delta(q, x, Z) \leq 1$ (tiene a lo sumo un movimiento a partir de cada configuración)
- $\delta(q, \lambda, Z) \leq 1$ (tiene a lo sumo un movimiento que no consuma la cadena a partir de cada configuración)
- Si $\delta(q, \lambda, Z) = 1$, entonces $\delta(q, x, Z) = \emptyset$

Teorema 6.5.1: No es cierto que para todo APD no determinístico exista otro determinístico que reconozca el mismo lenguaje

Teorema 6.5.2: Si L es un lenguaje regular, entonces existe un APD P determinístico tal que $\mathcal{L}(P) = L$

Demostración: La idea es sencilla, se construye un APD que esencialmente ignora la pila para simular un AFD. De manera más formal, sea A = $\langle Q, \Sigma, \delta, q_0, F \rangle$ un AFD, constrimos el APD determinístico P = $\langle Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F \rangle$ tal que:

- $\delta_P(q, x, Z_0) = \{(\delta(q, x), Z_0)\}$

Lo único que quedaría es probar por inducción sobre |w| que:

$$(q_0, w, Z_0) \overset{*}{\underset{P}{\vdash}} (p, \lambda, Z_0) \text{ sii } \hat{\delta}(q_0, w) = p$$

□

Definición 6.5.2: Un lenguaje L tiene la propiedad del prefijo sii para todo par de cadenas no nulas x e y, se cumple que

$$x \in L \Longrightarrow xy \notin L$$

Teorema 6.5.3: Un lenguaje L es $\mathcal{L}_\lambda(P)$ para algun APD determinístico P, si y solo si L tiene la propiedad del prefijo y existe algún APD determinísto P_2 tal que $\mathcal{L}(P_2) = L$

Teorema 6.5.4: Los lenguajes aceptados por APDs determinísticos incluyen a los regulares, y están incluidos en los libres de contexto

Teorema 6.5.5: Si L = $\mathcal{L}_\lambda(P)$, para algún APD determinístico P, entonces L tiene una GLC no ambigua

Demostración: El plan va a ser probar que la construcción utilizada en Teorema 6.4.3 produce una GLC no ambigua cuando se basa en APD determinístico. Primero, recordemos que por Teorema 5.6.1 tenemos que va a ser suficiente demostrar que tiene únicas derivaciones a izquierda para probar que G no es ambigua.

Supongamos entonces que P acepta la cadena w por pila vacía, tenemos que lo hace en una única secuencia de movimientos, pues es determinístico, por lo que podemos determinar qué producción permite que G derive a w.

Nunca va a haber más de una opción para qué cambio de configuración motivó la producción a utilizar, pero sí podemos tener más de una posible producción para un cambio dado. Por ejemplo, supongamos $\delta(q, \alpha, X) = \{(r, Y_1...Y_k)\}$, tenemos que para este movimiento tenemos todas las producciones en G, causada por todos los posibles órdenes en los que pueden estar los estados de P.

Sin embargo, solo una de estas producciones va a representar realmente los cambios realizados por P (pues si en algún momento tuviera más de una opción que lo llevara a aceptar w, el mismo no sería determinístico), luego, solo una de estas producciones realmente derivan en w. □

Teorema 6.5.6: Si L = $\mathcal{L}(P)$ para algun APD determinístico P, entonces L tiene una gramática no ambigua

Demostración: Hagamos uso de un símbolo especial \$ tal que no aparezca en las cadenas de L, y definamos L´ = L\$. Tenemos entonces que necesariamente L´ cumple la propiedad del prefijo, y, por Teorema 6.5.3, tenemos que existe un P´ tal que L´ = $\mathcal{L}_\lambda(P')$. Además, por Teorema 6.5.4 tenemos que existe una gramática G´ que genera el lenguaje de pila vacía de P´, es decir, L´.

Ahora construyamos un autómata G a partir de G´ tal que $\mathcal{L}(G) = L$. Para lograr esto, simplemente tenemos que deshacernos de los símbolos especiales \$ en las cadenas. Para lograr esto, podemos tratar estos símbolos como variables en las producciones de G, y hacer que sean la cabeza de una producción cuyo cuerpo es λ, o sea $\$ \to \lambda$.

El resto de las producciones se mantienen, luego, como $\mathcal{L}(G') = L'$, tenemos que $\mathcal{L}(G) = L$. Nos queda determinar no ambigüedad.

sin embargo, las derivaciones a izquierda de G son exactamente las mismas que para G´, con la excepción que al final de las derivaciones en G, se toma un paso más para deshacese del símbolo especial. Como tenemos que G´ es ambigua, G necesariamente también lo es.

□

7. Propiedades de Lenguajes Libres de Contexto

7.1. Formas normales de GLCs (Innecesario si repasando para final)

El contenido en esta sección (más allá de la mención de las formas normales), no se dictó en el segundo cuatrimestre de 2024 de la materia y es solo por interés personal

Definición 7.1.1: Llamamos forma normal de Chomsky a una producción la cual, siendo A, B, C variables, y a un símbolo terminal, es de alguna de las siguientes formas:

- $A \to BC$
- $A \to a$

El objetivo de esta sección va a ser demostrar que todo lenguaje libre de contexto (sin λ) generado por una GLC G, es generado por otra en cuyas producciones estén todas en forma normal de Chomsky. Para lograrlo, vamos a:

1. Eliminar símbolos inútiles
2. Eliminar producciones λ
3. Eliminar producciones unitarias

7.1.1. Eliminando símbolos inútiles

Definición 7.1.1.1: Decimos que un símbolo X es útil para una gramática G si hay alguna derivación tal que $s \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, para algún $w \in V_T^*$ (es decir, es parte de alguna forma sentencial). Notar que X puede o ser una variable o un símbolo terminal.

Si un símbolo no es útil, decimos que es inútil. Claramente, evitar estos símbolos no afecta el lenguaje generado por la gramática.

El proceso que vamos a utilizar para eliminar estos símbolos empieza por la identificación de dos propiedades que todo símbolo necesita para ser útil:

1. Decimos que un símbolo X está generando si se tiene que $X \overset{*}{\Rightarrow} w$, para alguna cadena w. Notar que todo terminal está generando, pues deriva en sí mismo en 0 pasos
2. Decimos que X es alcanzable si existe una derivación $S \Rightarrow \alpha X \beta$, para $\alpha, \beta \in (V_N \cup V_T)^*$

Teorema 7.1.1.1:

Sea G una GLC, tal que $\mathcal{L}(G) \neq \emptyset$; Sea $G_1 = \langle V_{N1}, V_{T1}, P_1, S_1 \rangle$ la gramática obtenida a partir de:

1. Eliminar todos los símbolos inútiles, y toda producción que contenga a uno o más de esos símbolos
2. Eliminar todos los símbolos no alcanzables en la gramática resultante del proceso anterior

Entonces G_1 no tiene símbolos inútiles, y $\mathcal{L}(G_1) = \mathcal{L}(G_2)$

Demostración: Supongamos, sin pérdida de generalidad, que X es un símbolo que no es eliminado, es decir, tenemos que $X \in (V_{T1} \cup V_{N1})$. Sea G_2 la gramática resultante de aplicar la primera regla

a G. Como X esta en G_1, necesariamente tenemos que $X \overset{*}{\underset{G_1}{\Rightarrow}} w$. Además, tenemos que cada símbolo usado en esta derivación también está generando, por lo que $X \overset{*}{\underset{G_2}{\Rightarrow}} w$.

Dado que X no fue eliminado al aplicar la segunda regla, tenemos que hay α, β tales que $S \overset{*}{\underset{G_2}{\Rightarrow}} \alpha X \beta$, y como todos estos símbolos son alcanzables, tenemos $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha X \beta$

Tenemos que todos los símbolos en $\alpha X \beta$ son alcanzables, y además, como todos están incluidos en $(V_{N2} \cup V_{T2})$, tenemos que están generando. Luego, una derivación del estilo $S \overset{*}{\underset{G_2}{\Rightarrow}} \alpha X \beta \overset{*}{\underset{G_2}{\Rightarrow}} xwy$ involucra símbolos alcanzables, pues son alcanzados por algún símbolo de $\alpha X \beta$, por lo que necesariamente se trata de una derivacíon que también se encuentra en G_1, es decir:

$$S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha X \beta \overset{*}{\underset{G_1}{\Rightarrow}} xwy$$

Concluimos entonces que X es un símbolo útil que se encuentra en G_1. Como X se trataba de un símbolo cualquiera, concluimos también que G_1 no tiene símbolos inútiles.

Ahora queda demostrar que $\mathcal{L}(G_1) = \mathcal{L}(G)$

- $\mathcal{L}(G_1) \subseteq \mathcal{L}(G)$

  Dado que simplemente eliminamos símbolos y producciones de G, es evidente (hará falta probar que eliminar símbolos y producciones de una gramática genera subconjuntos de su lenguaje?).

- $\mathcal{L}(G_1) \supseteq \mathcal{L}(G)$

  Tenemos que probar que si $w \in \mathcal{L}(G) \implies w \in \mathcal{L}(G_1)$. Sin embargo, que una cadena w esté en el lenguaje generado por G_1, implica necesariamente que $S \overset{*}{\underset{G}{\Rightarrow}} w$. Sin embargo, tenemos que todos los símbolos en esta derivación son necesariamente alcanzables y están generando, por lo que $S \overset{*}{\underset{G_1}{\Rightarrow}} w$

7.1.1.1. Computando símbolos alcanzables y generadores

Definición 7.1.1.1.1: Sea G una gramática, para computar los símbolos generadores de G hacemos uso de la siguiente inducción:

- Caso base: Todo símbolo en V_T es generador, pues se genera a sí mismo
- Paso inductivo:

  Sea $A \to \alpha$ una producción de G, y sea todo símbolo en α un símbolo que ya fue reconocido como generador, entonces A es un generador.

Teorema 7.1.1.1.1: El algoritmo propuesto solo encuentra todos los símbolos generadores

Demostración: En otras palabras, queremos demostrar que el algoritmo encuentra un símbolo X sii X es generador.

$\Longrightarrow$) Por como definimos el algoritmo, tenemos que necesariamente el símbolo encontrado debe ser generador (Se puede hacer induccion en el orden en el que se agragaron los símbolos para algo más formal, pero lo veo inecesario)

$\Longleftarrow$) Tenemos entonces que X es un generador, es decir, $X \underset{G}{\overset{i}{\Rightarrow}} w$. Probamos por recursión en el largo de la derivación que X es encontrado por el algoritmo.

- Caso base: i = 0

  Tenemos entonces que X $\in V_T$, por lo que es un generador de sí mismo

- Caso i = n, n > 0:

  Como n > 0, tenemos que X es una variable. Sea entonces la derivación $X \Rightarrow \alpha \overset{*}{\Rightarrow} w$, es decir, la primera produccion utilizada para la derivación es $X \to \alpha$. Tenemos que cada símbolo en α genera o an símbolo terminal de w, o a λ, además, cada una de estas derivaciones toma menos de i pasos, por lo que por H.I sabemos que el algoritmo los reconoce como generadores. Luego, en el paso inductivo del algoritmo, el mismo va a reconocer a X como generador al considerar la producción $X \to \alpha$

□

Definición 7.1.1.1.2: Sea G una gramática, para computar los símbolos alcanzables hacemos uso de la inducción:

- Caso base:

  S es alcanzable

- Paso inductivo:

  Supongamos que descubrimos que A es alcanzable, entonces tenemos que para todas las producciones con A en la cabeza, se cumple que todos los símbolos en ellas son alcanzables

Teorema 7.1.1.1.2: El algoritmo propuesto solo encuentre todos los símbolos alcanzables

7.1.2. Eliminando producciones λ

Definición 7.1.2.1: Decimos que una variable A es anulable si $A \underset{*}{\Rightarrow} \lambda$

Definición 7.1.2.2: Sea G una gramática, para computar los símbolos anulables aplicamos el siguiente algoritmo iterativo:

- Base:

  Si $A \to \lambda$ es una producción de G, entonces A es anulable

- Paso inductivo:

  Si hay una producción $B \to C_1...C_k$ donde cada C_i es anulable, entoces B es anulable

Teorema 7.1.2.1: El algoritmo encuentra un símbolo A sii es anulable

Demostración:

- $\Longrightarrow$) Por inducción en el orden en el que es encontrado
- $\Longleftarrow$) Por inducción en el mínimo i tal que $A \overset{i}{\to} \lambda$:
  - Caso base i = 1:

    En este caso tenemos que necesariamente $A \to \lambda$ es una producción de G, por lo que es descubierta en el paso base

  - Caso i = n, n $>$ 1:

    Tenemos que $A \overset{n}{\Rightarrow} \lambda$. El primer paso de la derivación debe ser del estilo $A \Rightarrow C_1...C_k \overset{n-1}{\Rightarrow} \lambda$. Tenemos entonces que cada C_i deriva en λ en menos de n pasos, por lo que, por H.I, tenemos que el algoritmo lo identifica como anulable. Finalmente, el algoritmo determina a A como símbolo anulable mediante la producción $A \to C_1...C_k$

□

Teorema 7.1.2.2: Sea G = $\langle V_N, V_T, P, S\rangle$ una GLC, existe una GLC $G' = \langle V_N, V_T, P', S\rangle$ tal que G´ no tenga símbolos inútiles y $\mathcal{L}(G') = \mathcal{L}(G) - \{\lambda\}$

Demostración: Primero definimos P´ (pues es la única modificación en comparación a G).

Una vez identificados los símbolos inútiles, definimos P´ tal que:

- Por cada producción $A \to X_1X_2...X_k$, con $k \geq 1$, sea m la cantidad de símbolos anulables en el cuerpo de la producción, entonces P´ va a tener 2^m versiones de esta producción, una por cada posible combinación en la que uno o más de los símbolos anulables no están presente (representando así el hecho de haber tomado estas producciones en G, y luego seguir hasta llegar a λ)
- Si tenemos que m = k, (todos los símbolos son anulables), entonces va a haber una menos, pues el caso en el que se tomaron todas hasta llegar a λ no está incluido

Queremos ahora probar que $w \in \mathcal{L}(G') \iff w \in \mathcal{L}(G) - \{\lambda\}$. Para ello. vamos a probar primero que:

$$A \overset{*}{\underset{G´}{\Rightarrow}} w \text{ sii } A \overset{*}{\underset{G}{\Rightarrow}} w \text{ y } w \neq \lambda$$

- $\Longrightarrow$)

  Tenemos $A \overset{i}{\underset{G´}{\Rightarrow}} w$, por lo que necesariamente $w \neq \lambda$ ya que G´ no tienen producciones λ. Probamos ahora por inducción sobre el largo de la derivación que $A \overset{*}{\underset{G}{\Rightarrow}} w$:

  - Caso base i = 1: En este caso tenemos que hay una producción $A \underset{G´}{\rightarrow} w$ en P´. Por como construimos G´, tenemos que G debe tener una producción $A \rightarrow \alpha$ tal que los símbolos de w se encuentren en el cuerpo de la producción, y que haya cero o más símbolos anulables entre ellos, por lo que tenemos que $A \underset{G}{\Rightarrow} \alpha \overset{*}{\underset{G}{\Rightarrow}} w$ (el resto de los símbolos derivan en λ)
  - Caso i = n, n $>$ 1: Tenemos entonces que la derivación es de la forma $A \underset{G´}{\Rightarrow} X_1...X_k \overset{*}{\underset{G´}{\Rightarrow}} w$. Por definición de P´, tenemos que la primera producción utlizada para esta derivación debe haber sido construida a partir de una producción en G del estilo $A \rightarrow Y_1...Y_j$, donde la secuencia de Ys son los Xs, ordenados, pero con cero o mas símbolos anulables entre sí.

    Sea entonces $w = w_1...w_k$ una descomposición de w tal que $X_l \overset{*}{\underset{G´}{\Rightarrow}} w_l$. Tenemos que o bien X_l es una variable, o bien es un terminal, pero en ambos casos se puede aplicar la H.I, por lo que sabemos que $X_l \overset{*}{\underset{G}{\Rightarrow}} w_l$.

    Con esto, podemos construir la siguiente derivación en G:

    $$A \underset{G}{\Rightarrow} Y_1...Y_j \overset{*}{\underset{G}{\Rightarrow}} X_1...X_k \underbrace{\overset{*}{\underset{G}{\Rightarrow}}}_{H.I} w_1...w_k = w$$

    Donde el primer paso de la derivación proviene del uso de la priducción $A \rightarrow Y_1...Y_j$, que por lo argumentado anteriormente sabemos que existe. El siguiente paso proviene de la derivación en λ de todos los símbolos anulables que no son ningunos de los X_l, y el último paso proviene de la derivación de estos símbolos en w_l, que sabemos está en G también por la H.I

- $\Longleftarrow$) Tenemos que $A \overset{i}{\underset{G}{\Rightarrow}} w$ y $w \neq \lambda$. Probamos ahora por inducción en i que esto implica $A \overset{*}{\underset{G´}{\Rightarrow}} w$
- Caso base i = 1: En este caso tenemos que $A \rightarrow w$ necesariamente es una producción de G y que, como $w \neq \lambda$, también se encuentra en G´, por lo que $A \underset{G´}{\Rightarrow} w$
- Caso i = n, n > 1: Tenemos entonces que la derivación es del estilo $A \underset{G}{\Rightarrow} Y_1...Y_m \overset{*}{\underset{G}{\Rightarrow}} w$. Sea entonces $w = w_1...w_m$ tal que $Y_j \overset{*}{\underset{G}{\Rightarrow}} w_j$.

  Sean $X_1...X_k$ aquellos Y_j s, en orden, tales que $w_j \neq \lambda$. Tenemos que por definición de G´ que tiene una producción $A \rightarrow X_1...X_k$. Necesariamente $X_1...X_k \overset{*}{\underset{G}{\Rightarrow}} w$, pues aquellas variables que ahora no están presente originalmente derivaban en λ y, como cada una de las derivaciones $X_l \overset{*}{\Rightarrow} w_l$, toma menos de n pasos, podemos usar la H.I para concluir que:

  $$A \underset{G´}{\Rightarrow} X_1...X_k \underbrace{\overset{*}{\underset{G´}{\Rightarrow}}}_{H.I} w_1...w_k = w$$

Terminamos la demostración reemplazando A por S:

$$w \in \mathcal{L}(G_1) \Longleftrightarrow S \overset{*}{\underset{G´}{\Rightarrow}} w$$

$$\Longleftrightarrow S \overset{*}{\underset{G}{\Rightarrow}} \wedge\, w \neq \lambda \Longleftrightarrow w \in \mathcal{L}(G) - \{\lambda\}$$

□

7.1.3. Eliminando producciones unitarias

Definición 7.1.3.1: LLamamos producción unitaria a aquellas producciones de la forma $A \to B$ tal que A y B son variables

Definición 7.1.3.2: LLamamos parejas unitarias a todas las parejas de variables (A, B) tales que $A \overset{*}{\Rightarrow} B$ usando solo producciones unitarias. Inductivamente:

- Base: (A, A) es una pareja unitaria para cualquier variable A
- Inductivo: Sea (A, B) una pareja que ya se determinó es unitaria, y sea $B \to C$ una producción, con C una variable, entonces (A, C) es una pareja unitaria

Teorema 7.1.3.1: Sea G una GLC, el algoritmo identifica una pareja unitaria (A, B) sii $A \underset{G}{\overset{*}{\Rightarrow}} B$ usando solo producciones unitarias

Demostración:

- $\Longrightarrow$) inducción en el orden en el que son agragados
- $\Longleftarrow$) Tenemos que $A \underset{G}{\overset{i}{\Rightarrow}} B$, demostramos por inducción sobre i que el algoritmo encuentra la pareja (A, B)

  ▸ Caso base i = 0: Tenemos $A = B$, por lo que la pareja (A, A) es identificada en el caso base

  Caso i = n, n > 0: Tenemos $A \underset{G}{\overset{n}{\Rightarrow}} B$ solo haciendo uso de producciones unitarias, en particular, tenemos que la derivación es de la forma:

  $$A \underset{G}{\overset{n-1}{\Rightarrow}} C \Rightarrow B$$

  La derivación $A \underset{G}{\overset{n-1}{\Rightarrow}} C$ toma menos de n pasos y solo usa producciones unitarias, por lo que, por H.I, tenemos que el algoritmo identifica la pareja (A, C). Luego, como necesariamente tiene que estar la producción $C \to B$ en la gramática, el algoritmo identifica por el caso recursivo la pareja (A, B)

□

Teorema 7.1.3.2: Sea G una GLC, existe una GLC $G_1 = \langle V_N, V_T, P_1, S \rangle$ sin producciones unitarias tal que $\mathcal{L}(G) = \mathcal{L}(G_1)$

Demostración: Definimos P_1 tal que no tenga producciones unitarias, para cada pareja (A, B), si $B \to \alpha$ no es una producción unitaria de P, entonces $A \to \alpha$ es una producción de P_1.

Ahora queda demostrar que $w \in \mathcal{L}(G) \iff w \in \mathcal{L}(G_1)$

- $\Longleftarrow$) Tenemos $S \underset{G_1}{\overset{*}{\Rightarrow}} w$. Como cada producción en G_1 es quivalente a una secuencia de zero o más producciones unitarias en G, tenemos que $\alpha \underset{G_1}{\overset{*}{\Rightarrow}} \beta$ implica que $\alpha \underset{G}{\overset{*}{\Rightarrow}} \beta$. Es decir, toda paso en una derivación en G_1 puede ser reemplazado por uno o más pasos en G, luego $S \underset{G}{\overset{*}{\Rightarrow}} w$
- $\Longrightarrow$) Tenemos que $w \in \mathcal{L}(G)$, por lo que sabemos que w tiene una derivación a izquierda, es decir $S \underset{L}{\Rightarrow} w$. Cada vez que se hace uso de una producción unitaraia para un paso de la derivación, tenemos que la única variable en el cuerpo de la priducción se vuelva la variable más a la izquierda, por lo que es inmediatamente reemplazada. Por lo que podemos separar una derivación en G en una secuencia de pasos donde 0 o más producciones unitarias son seguidas por una no unitaria. Sin embargo, tenemos que son estos mismos pasos los simulados por G_1, , ya que la construcción de P_1 salta estas derivaciones "intermedias" entre las producciones no unitarias. Por lo que tenemos que $S \underset{G_1}{\overset{*}{\Rightarrow}} w$

□

7.1.4. Forma normal de Chomsky

Definición 7.1.4.1: Recordando la definición de FNC, tenemos que se trata de una GLC G tal que todas sus producciones estén en una de dos formas:

1. $A \to BC$, con todos los símbolos siendo variables
2. $A \to a$, con A una variable y a un terminal

Además, G no tiene símbolos inútiles

Teorema 7.1.4.1: Si G es una GLC que contiene al menos una cadena además de λ, entonces hay otra CFG G_1 tal que la misma no tiene, producciones lambda, producciones unitarias, ni símbolos inútiles, y se cumple que $\mathcal{L}(G_1) = \mathcal{L}(G) - \{\lambda\}$

Demostración: Aplicamos, en orden, los siguientes pasos:

1. Eliminamos producciones λ
2. Eliminamos producciones unitarias
3. Eliminamos símbolos inútiles

□

Teorema 7.1.4.2: Si G es una GLC cuyo lenguaje contiene al menos una cadena además de λ, existe una gramática G_1 en FNC tal que $\mathcal{L}(G_1) = \mathcal{L}(G) - \{\lambda\}$

Demostración: Primero usamos Teorema 7.1.4.1 para conseguir una gramática G_2 que no tenga producciones λ, producciones unitarias, ni símbolos inútiles, y que cumpla que $\mathcal{L}(G_2) = \mathcal{L}(G) - \{\lambda\}$. Con esto ahora construimos G_1 de la siguiente manera:

1. Por cada símbolo terminal a en una producción cuyo cuerpo tenga longitud mayor a 2, creamos una nueva variable, A, y agregamos la producción $A \to a$, reemplazando todas las apariciones previas de a por esta nueva variable
2. Por cada producción $A \to B_1...B_k$, con k $\geq$ 3 (observar que son todas variables por el paso anterior), las dividimos en un grupo de particiones introduciendo k - 2 nuevas variables $C_1...C_{k-2}$, y reemplazamos la producción original por k - 1 producciones nuevas tal que se tenga:

$$A \to B_1C_1, \quad C_1 \to B_2C_2, \quad ... \quad C_{k-2} \to B_{k-1}B_k$$

Queremos ahora probar que $w \in \mathcal{L}(G_2) \iff w \in \mathcal{L}(G_1)$

- $\Longrightarrow$) Si w tiene una derivación en G_2, podemos reemplazar las producciones utilizadas en un paso arbitrario, digamos $A \to X_1...X_k$ con una secuencia de producciones en G_1.

  Si X_i es un terminal, entonces tenemos que G_1 tiene una variable B_i, tal que $B_i \to B_i$. Si k > 2 tenemos que G_1 tiene una secuencia de producciones $A \to B_1C_1, \quad C_1 \to B_2C_2...$ donde cada B_i es o bien la variable introducida para representar al terminal X_i si el mismo era un terminal, o el mismo X_i, si se trataba de una variable en G_2. Como esta secuencia de producciones simulan cualquier producción utilizada en algún paso de la derivación de w en G_2, tenemos que $w \in \mathcal{L}(G_1)$
- $\Longleftarrow$)

□

7.2. Lema de Pumping para Lenguajes Libres de Contexto

Lema 7.2.1: Sea $G = \langle V_N, V_T, P, S\rangle$ una GLC con P $\neq \emptyset$, sea $\alpha \in (V_N \cup V_T)^*$ y sea $\mathcal{T}(S)$ un árbol de derivacíon con altura h. Sea:

$$a = \max\{k : (k = |\beta|, A \to \beta \in P, \beta \neq \lambda) \ \ o \ \ (k = 1, A \to \lambda \in P)\}$$

Entonces $a^h \geq |\alpha|$. (Informalmente, como mucho podemos haber usado la produccion con el cuerpo más largo en cada paso de la derivación)

Demostración: Por inducción en h.

- Caso base, h = 0:

  El único árbol de derivación posible es aquél en el que el único nodo es S, por lo que tenemos $a^0 = 1 = |S|$

- Paso inductivo:

  Sea γ la base del árbol de altura h (es decir, su producción), tenemos que, por H.I: $a^h \geq \gamma$. Sea α entonces el producto Del árbol de altura h + 1.

  Falta dibujito

  Tenemos que, como a lo sumo cada símbolo de γ pudo haber hecho uso de la produccion más larga (es decir, de longitud a) para seguir con la derivación, resulta que:

$$a\ |\gamma| \geq |\alpha|$$

Por H.I, tenemos $a^h \geq |\gamma|$, por lo que

$$a^{h+1} = aa^h \geq a\ |\gamma| \geq |\alpha|$$

□

Definición 7.2.1: Para todo lenguaje libre de contexto L, existe n > 0 tal que para toda cadena $\alpha \in L$ con $|\alpha| \geq n$ se cumple que:

- Existe una descomposición de α tal que $\alpha = rxyzs$
- $|xyz| \leq n$
- $|xz| \geq 1$
- Para todo $i \geq 0$, la cadena rx^iyz^is pertenece a L.

Demostración: Sea G una GLC tal que L = $\mathcal{L}(G)$ y sea b = $\max\{k : (k = |\beta|, A \to \beta \in P, \beta \neq \lambda)\ o\ (k = 1, A \to \lambda \in P)\}$

- Caso $a = 1$: En este caso, tenemos que las únicas producciones de G solo consisten o de un sólo terminal, o de λ, o se tratan de producciones unitarias. Sin importar el caso, tenemos que la cadena resultante de cualquier derivación en esta gramática tendrá longitud a lo sumo 1, por lo que con n ≥ 2 tenemos que el antecedente del teorema es falso, por lo que el teorema en sí es trivialmente verdadero.
- Caso $a \geq 2$:

  Tomemos $n = a^{|V_N|+1}$, y consideremos la cadena α tal que $|\alpha| \geq n$. Sea $\mathcal{T}(S)$ unrbol mínimo de derivación para α. Tenemos que, por el lema anterior, que la altura del árbol debe ser al menos $|V_N| + 1$, pues:

  $$a^h \geq |\alpha| \geq n = a^{|V_N|+1} \iff h \geq |V_N| + 1$$

  Luego, necesariamente debe haber un camino hacia alguno de los símbolos de α producidos, uno de longitud $h \geq |V_N| + 1$, pero como solo hay $|V_N|$ variables, tenemos que debe haber alguno repetido.

  Llamemos A a alguna de estas variables que se repiten en el camino, en particular, escogemos la primera que aparezca recorriendo el árbol de manera ascendente:

  Acá falta dibujito :(

  La segunda aparición de A da lugar a la cadena xyz. Tenemos, además, que esta segunda aparición debe estar a una altura h´ $\leq |V_N| + 1$ de la base (por que a esa altura necesariamente tiene que haber ocurrido una repetición de alguna variable), por lo que:

  $$|\ xyz\ | \leq a^{h´} \leq a^{|V_N|+1} = n$$

  Cumpliendo así la primera condición.

  Cuando las cadenas x y z son simultáneamente nulas, tenemos que podemos reemplazar la segunda aparición de A por la primera, «compactando» el árbol, y seguir generando a α, sin

embargo, dijimos que $\mathcal{T}(S)$ era el árbol mínimo que generaba α, por lo que llegamos a un absurdo. El absurdo vino de suponer que z y x podían ser simultanemente nulas, por lo que con esto tenemos la segunda condición.

Falta dibuuuuu

Finalmente, demostremos que $\forall i \geq 0, rx^iyz^is \in L$, por inducción en i:

- Caso base i = 0:

  Tenemos que $S \overset{*}{\Rightarrow} rAs \overset{*}{\Rightarrow} rys$, por lo que necesariamente $rys = rx^0yz^0s \in L$

- Caso i > 0: Tenemos por H.I que $rx^iyz^is \in L$, veamos que vale para i + 1.

  Sabemos que $S \overset{*}{\Rightarrow} rx^iAz^is \overset{*}{\Rightarrow} rx^iyz^is$, pero también tenemos que, en vez de hacer que A tome el camino de producciones que deriva en y, puede volver a derivar xAz, es decir:

  $$S \overset{*}{\Rightarrow} rx^iAz^is \overset{*}{\Rightarrow} rx^ixAzz^is \overset{*}{\Rightarrow} rx^{i+1}yz^{i+1}s$$

  Por lo que $rx^{i+1}yz^{i+1}s \in L$

□

7.3. Propiedades de clausura de Lenguajes Libres de Contexto (Link a Demos más formales°)

7.3.1. Unión

Teorema 7.3.1.1: Si L_1 y L_2 son lenguajes libres de contexto, $L_1 \cup L_2$ también lo es

Demostración: Como ambos lenguajes son libres de contexto, entonces existen G_1 y G_2 GLCs tales que $\mathcal{L}(G_1) = L_1$ y $\mathcal{L}(G_2) = L_2$. Supogamos, sin pérdida de generalidad, que $V_{N_1} \cap V_{N_2} = \emptyset$ Definimos entonces la gramática G = $\langle V_{N_1} \cup V_{N_2} \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S\rangle$

Puede demostrarse fácilmente por inducción sobre el largo de la cadena w que $w \in \mathcal{L}(G) \iff w \in \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$

7.3.2. Concatenación

Teorema 7.3.2.1: Si L_1 y L_2 son lenguajes libres de contexto, L_1L_2 también lo es

Demostración: Como ambos lenguajes son libres de contexto, entonces existen G_1 y G_2 GLCs tales que $\mathcal{L}(G_1) = L_1$ y $\mathcal{L}(G_2) = L_2$. Supogamos, sin pérdida de generalidad, que $V_{N_1} \cap V_{N_2} = \emptyset$ Definimos entonces la gramática G = $\langle V_{N_1} \cup V_{N_2} \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1S_2\}, S\rangle$

Puede demostrarse fácilmente por inducción sobre el largo de la cadena w que $w \in \mathcal{L}(G) \iff w \in \mathcal{L}(G_1)\mathcal{L}(G_2)$ □

7.3.3. Clausura de Kleene

Teorema 7.3.3.1: Si L es un lenguaje libre de contexto, L^* también lo es

Demostración: Como L es un lenguaje libre de contexto, existe G GLC tal que $\mathcal{L}(G) = L$. Sea entonces G´ = $\langle V_N, \Sigma, P \cup \{S \to SS' \mid \lambda\}, S' \rangle$

Solo queda demostarr que $w \in \mathcal{L}(G) \iff w \in L^*$ □

7.3.4. Reversa

Teorema 7.3.4.1: Si L es un lenguaje libre de contexto, también lo es L^R

Demostración: Sea G una GLC cuyo lenguaje es L, construimos $G^R = \langle V_N, \Sigma, P^R, S \rangle$ donde P^R es el "reverso" de cada producción en P. De manera más específica, si $A \to \alpha$ es una producción en G, entonces $A \to \alpha^R$ lo es en G^R.

Queda demostrar, por inducción en la cantidad de pasos de la derivación, que $\mathcal{L}(G^R) = L^R$ □

7.3.5. Intersección

Teorema 7.3.5.1: Los lenguajes libres de contexto no están cerrados respecto de la intersección

Demostración: Tenemos que $L = \{0^n 1^n 2^n \mid n \geq 1\}$ no es libre de contexto, pero $L_1 = \{0^n 1^n 2^i \mid n, i \geq 1\}$ y $L_2 = \{0^i 1^n 2^n \mid n, i \geq 1\}$ lo son. □

7.3.6. Complemento

Teorema 7.3.6.1: Los lenguajes libres de contexto no son cerrados respecto del complemento

Demostración: Si lo fueran, entonces $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ lo sería □

7.3.7. Diferencia

Teorema 7.3.7.1: L_1 - L_2 no es necesariamente libre de contexto

Demostración: Si lo fuera, entonces, como sabemos que Σ^* es libre de contexto, deberíamos tener que $\Sigma^* - L = \overline{L}$ es libre de contexto □

7.3.8. Intersección con un lenguaje regular

Teorema 7.3.8.1: Si L es un lenguaje libre de contexto, y R uno regular, entoces $L \cap R$ es libre de contexto

Demostración: Sea $P = \langle Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P \rangle$ un autómata de pila para L, y $M = \langle Q_M, \Sigma, \delta_A, q_A, F_A \rangle$ un AFD para R, construimpos el APD
$P' = \langle Q_P \times Q_M, \Sigma, \Gamma, \delta, (q_P, q_A), Z_0, F_P \times F_A \rangle$, donde:

- $\delta((q,p),a,X) = \left\{((r,s),\gamma) : s = \hat{\delta}_A(p,a) \wedge (r,\gamma) \in \delta_P(q,a,X)\right\}$

Queda demostrar, por inducción sobre la cantidad de movimientos, que $(q_P, w, Z_0) \overset{*}{\underset{P}{\vdash}} (q, \lambda, \gamma) \iff ((q_P, q_A), w, Z_0) \overset{*}{\underset{P'}{\vdash}} \left(\left(q, \hat{\delta}_A(q_A, w)\right), \lambda, \gamma\right)$.

□

7.4. Problemas de decisibilidad para Lenguajes Libres de contexto

- Determinar si un Lenguaje es vacío se puede hacer en orden lineal respecto al tamaño de la representación
- Determinar si una cadena pertenece a un lenguaje se puede hacer en orden cúbico
- Determinar si un lenguaje libre de contexto es finito o infinito
- Los siguientes problemas no son decidibles:
  - Determinar si una GLC es ambigua
  - Determinar si un Lenguaje libre de contexto es inherentemente ambiguo
  - DEterminar si la intersección de dos lenguajes libres de contexto es ambigua
  - Determinar si dos lenguajes libres de contexto son iguales
  - Determinar si un lenguaje libre de contexto es Σ^*

8. APDs y LCs Determinísticos

Recordemos la definición de un APD determinístico

Definición 8.1: Un autómata de pila $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, con:

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$$

es determinístico si para cada $a \in \Sigma, q \in Q, Z \in \Gamma$:

- $\delta(q, a, Z)$ contiene a lo sumo un elemento y $\delta(q, \lambda, Z)$ es $\emptyset$

O

- $\delta(q, a, Z) = \emptyset$, y $\delta(q, \lambda, Z)$ contiene a lo sumo un elemento

Para facilitar la notación, vamos a escribir $\delta(q, a, Z) = (r, \gamma)$ en vez de $\delta(q, a, Z) = \{(r, \gamma)\}$

8.1. APDs determinísticos capaces de leer toda la cadena

Lema 8.1.1: Sea $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ un APD determinístico. Es posible construir un APD determinístico equivalente P´ tal que:

1. Para todo $a \in \Sigma, q \in Q´, Z \in \Gamma´$:
   a) O bien $\delta´(q, a, Z)$ contiene exactamente un elemnete y $\delta´(q, \lambda, Z) = \emptyset$
   b) O bien $\delta´(q, a, Z) = \emptyset$ y $\delta´(q, \lambda, Z)$ contiene exactamente un elemento
2. Si $\delta´(q, a, Z_0´) = (r, \gamma)$, entonces $\gamma = \alpha Z_0´, \alpha \in \Gamma *$

Demostración: La idea es usar como marcador a $Z_0´$ para evitar que se vacíe la pila. Sea $\Gamma´ = \Gamma \cup \{Z_0´\}, Q´ = Q \cup \{q_0´, q_\lambda\}.\delta´$ queda definida como:

1. $\delta´(q_0´, \lambda, Z_0´) = (q_0, Z_0 Z_0´)$
2. $\forall q \in Q, Z \in \Gamma, a \in \Sigma \cup \{\lambda\}$ tales que $\delta(q, a, Z) \neq \emptyset$, tenemos que $\delta´(q, a, Z) = \delta(q, a, Z)$
3. Si $\delta(q, \lambda, Z) - \emptyset$ y $\delta(q, a, Z) = \emptyset$, entonces $\delta´(q, a, Z) = (q_\lambda, Z)$
4. $\forall a \in \Sigma, Z \in \Gamma´, \delta´(q_\lambda, a, Z) = (q_\lambda, Z)$

Queda probar que $\mathcal{L}(P) = \mathcal{L}(P´)$ □

8.2. Configuraciones ciclantes

Definición 8.2.1: Decimos que una configuración (q, w, a) de un APD determinístico P está en un ciclo si, $\forall i, i \geq 1$, esiste una configuración (p_i, w, β_i) tal que $|\beta_i| \geq |\alpha|$ y:

$$(q, w, \alpha) \vdash (q_1, w, \beta_1) \vdash (q_2, w, \beta_2) \ldots$$

Es decir, una configuración está en un ciclo si P puede hacer una cantidad infinita de movimientos λ sin reducir el tamaño de la pila

Definición 8.2.2: El siguiente algoritmo detecta configuraciones ciclantes:

Input: Un APD deterministico P = $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$

Output: Dado por los siguientes dos conjuntos
1. $C_1 = \Big\{(q, A) \mid (q, \lambda, A)$ es una configuración ciclante, y $\nexists r \in F$ tal que $(q, \lambda, A) \overset{*}{\vdash} (r, \lambda, \alpha)\Big\}$
2. $C_2 = \{(q, A) \mid (q, \lambda, A)$ es una configuración ciclante y $(q, \lambda, A) \vdash (r, \lambda, \alpha)$, para algún estado final $r\}$

Método: Sea $\#Q = n_1, \#\Gamma = n_2, y$ l es la longitud de la cadena más larga que P puede escribir en la pila en un sólo movimiento. Sea $n_3 = n_1 \frac{n_2^{n_1 n_2 l} - n_2}{n_2 - 1}$. n_3 es la máxima cantidad de movimientos λ que P puede hacer sin ciclar.

1. Por cada $q \in Qy A \in \Gamma$ determinar si $(q, \lambda, A) \overset{n_3}{\vdash} (r, \lambda, \alpha)$,para algún $r \in Q, \alpha \in \Gamma^+$. Caso positivo, (q, λ, A) es una configuración ciclante
2. Si (q, λ, A) es una configuración ciclante, determinar si existe algún $r \in F$ tal que $(q, \lambda, A) \overset{j}{\vdash} (r, \lambda, \alpha), 0 \leq j \leq n_3$. Caso positivo, agregarlo a C_2, caso negativo, a C_1

Teorema 8.2.1: El algoritmo correctamente determina C_1 y C_2

Demostración: **Pendiente** □

8.3. APDs determinísticos continuos

Definición 8.3.1: Un APD deterministico P = $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ es continuo si para todo $w \in \Sigma^*, \exists p \in Q, \alpha \in \Gamma^*$ tal que $(q_0, w, Z_0) \overset{*}{\vdash} (p, \lambda, \alpha)$. En otras palabras, se trata de un autómata que siempre consume toda la cadena de entrada

Lema 8.3.1: Sea P = $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ un APD determinístico, existe otro P´ tal que el mismo sea equivalente y continuo

Demostración: Supongamos, por Lema 8.1.1, que P siempre tiene un siguiente movimiento. Sea $P' = (Q \cup \{p, r\}, \Sigma, \Gamma, \delta', q_0, Z_0, F \cup \{p\})$, con δ' definida por:

1. $\forall q \in Q, a \in \Sigma, Z \in \Gamma, \delta'(q, a, Z) = \delta(q, a, Z)$
2. $\forall q \in Q, a \in \Sigma, Z \in \Gamma$ tales que (q, λ, Z) no sea una configuración ciclante, entonces $\delta'(q, \lambda, Z) = \delta(q, \lambda, Z)$
3. Para todo $(q, Z) \in C_1, \delta'(q, \lambda, Z) = (r, Z)$
4. Para todo $(q, Z) \in C_2, \delta'(q, \lambda, Z) = (p, Z)$
5. Para todo $a \in \Sigma, Z \in \Gamma, \delta'(p, a, Z) = (r, Z)$ y $\delta'(r, a, Z) = (r, Z)$

De esta manera, P´ simula P. Si se entra a una configuración ciclante en P cuyo ciclo pasaba por un estado final, ahora va a ir a parar directamente a p, (desde donde si todavía no consumió toda la cadena va al estado r) caso contrario va a r directamente, donde va a terminar de consumir la cadena.

□

8.4. Clausura de APDs determinísticos bajo complemento

Teorema 8.4.1: Si L = $\mathcal{L}(P)$ para algún APD determinístico P, entonces $\overline{L} = \mathcal{L}(P')$ para algún APD determinístico P´

Demostración: Sea P = $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ continuo. Definimos
P´ = $\langle Q', \Sigma, \Gamma, \delta', {q_0}', Z_0, F'\rangle$ donde:

$$\text{Q´ = \{[q, k]: q in Q and k in \{0,1,2\}\}}$$

El propósito de k es indicar si entre transiciones con consumo de entrada en P pasó o no por un estado final

$$q_0 = \begin{cases} [q_0,0] \text{ si } q_0 \notin F \\ [q_0,1] \text{ si } q_0 \in F \end{cases}$$

$$F' = \{[q, 2] : q \in Q\}$$

0 indica que P no pasó por F

1 indica que P sí paso por F

2 indica que P no pasó por F y P va a seguir leyendo

Para todo $q \in Q$, $[q, 2]$ es un estado final al que llega P´ antes que P lea un nuevo símbolo

La función de transición δ' queda definidda por:

- Si P lee desde q un símbolo $a, \delta(q, \lambda, Z) = \emptyset, y, \delta(q, a, Z) = (p, \gamma)$ entonces

  $\delta'([q, 0], \lambda, Z) = ([q, 2], Z)$

  P´ acepta la entrada antes de leer a porque P no la aceptó,

  $$\delta'([q, 1], a, Z) = \delta'([q, 2], a, Z) = \begin{cases} ([p,0],\gamma) \text{ si } p \notin F \\ ([p,1],\gamma) \text{ si } p \in F \end{cases}$$

- Si P no lee desde q símbolo $a, \delta(q, \lambda, Z) = (p, \gamma), y, \delta(q, a, Z) = \emptyset$ entonces:

  $$\delta'([q, 1], \lambda, Z) = ([p, 1], \gamma)$$

  $$\delta'([q, 0], \lambda, Z) = \begin{cases} ([p,0],\gamma) \text{ si } p \notin F \\ ([p,1],\gamma) \text{ si } o \in F \end{cases}$$

Para terminar, ahora queda ver que $\delta(q_0, w, Z_0) \in$

□

9. Máquinas de Turing

Por ahora solo voy a dejar los teoremas y lemas sin probarlos

Definición 9.1: Una Máquina de Turing (MT) es una tupla $M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$ donde:

- Q es el conjunto finito de estados del controlador finito (o cabeza)
- Σ es el conjunto finito de símbolos de entrada
- Γ es el conjunto de símbolos de cinta; Σ es siempre un subconjunto de Γ
- δ es la función de transición. Los argumentos de $\delta(q, X)$ son un estado q y un símbolo de cinta X y su valor, si está definido, es una tripla (p, Y, D) tal que:
  - $p \in Q$ es el siguiente estado
  - $Y \in \Gamma$ es el ßimbolo escrito en la celda siendo escaneada, reemplazando cualquier símbolo que estuviese ahí
  - D es una dirección, L o R, denotando "derecha" o "izquierda", indicando la dirección en la que se mueve la cabeza
- $q_0 \in Q$ es el estado inicial
- $B \in \Gamma, \notin \Sigma$ es el símbolo denotando una celda blanca
- $F \subset Q$ es el conjunto de estados finales

10. Ejercicios (Después debería pasarlo a otro lado)

1. Demostra que $\hat{\delta}(q, xy) = \hat{\delta}\Big(\hat{\delta}(q, x), y\Big)$ para cualquier estado q y cadenas x e y. **Pista**: hacer inducción sobre |y|

   Demostración: Siguiendo la sugerencia:

   - Caso base: |y| = 0 (y = λ)

   $\hat{\delta}(q, x\lambda) = \hat{\delta}\Big(\hat{\delta}(q, x), \lambda\Big) \underset{\text{def } \hat{\delta}}{=} \hat{\delta}(q, x)$ $\underset{\lambda \text{ es neutro}}{=}$ $\hat{\delta}(q, x\lambda)$

   - Paso inductivo:

   qvq $\hat{\delta}(q, xy) = \hat{\delta}\Big(\hat{\delta}(q, x), y\Big)$. Sea entonces $y = y'\alpha$, tenemos, reescribiendo el lado derecho de la igualdad, $\hat{\delta}\Big(\hat{\delta}(q, x), y'\alpha\Big) \underset{\text{def de } \hat{\delta}}{=} \delta\Big(\hat{\delta}\Big(\hat{\delta}(q, x), y'\Big), \alpha\Big) \underset{\text{H.I}}{=} \delta\Big(\hat{\delta}(q, xy'), \alpha\Big) \underset{\text{def de } \hat{\delta}}{=}$ $\hat{\delta}(q, xy'\alpha) = \hat{\delta}(q, xy)$. □

2. Demostrá que para cualquier estado q, cadena x, y símbolo α, $\hat{\delta}(q, \alpha x) = \hat{\delta}(\delta(q, \alpha), x)$

   Demostración: Por el ejercicio anterior, tenemos que $\hat{\delta}(q, \alpha x) = \hat{\delta}\Big(\hat{\delta}(q, \alpha), x\Big)$. Luego, solo resta probar que $\hat{\delta}(q, \alpha) = \delta(q, \alpha)$ que sale fácil usando la definición □

3. Sea M un AFD y q un estado del mismo, tal que $\delta(q, \alpha) = q$ para cualquier símbolo α. Demostrar por inducción sobre el largo de la cadena de entrada que para toda cadena ω, $\hat{\delta}(q, \omega) = q$

   Demostración: Haciendo inducción sobre el largo de la cadena w:

   - Caso $|w| = 0$ ($w = \lambda$)

     $\hat{\delta}(q, \lambda) \underset{\text{def } \hat{\delta}}{=} \text{q}$

   - Caso $w = xa$

   $\hat{\delta}(q, xa) \underset{\text{def } \hat{\delta}}{=} \delta\Big(\hat{\delta}(q, x), a\Big) \underset{HI}{=} \delta(q, a) \underset{\text{Por enunciado}}{=} q$

   □

4. Sea M un AFD y α un símbolo particular de su alfabeto, tal que para todos los estados q de M tenemos que $\delta(q, \alpha) = q$

   a) Demostrar por inducción sobre n que para cualquier $n \geq 0, \hat{\delta}(q, \alpha^n) = q$

Demostración: Haciendo inducción sobre n:

- Caso $n = 0$

  $\hat{\delta}(q, a^0) = \hat{\delta}(q, \lambda) \underset{(\text{def } \hat{\delta})}{=} \text{q}$
- Caso $n \geq 1$

$\hat{\delta}(q, a^n) \underset{\text{def } \hat{\delta}}{=} \delta\Big(\hat{\delta}(q, a^{n-1}), a\Big) \underset{HI}{=} \delta(q, a) \underset{\text{Por enunciado}}{=} q$

□

b) Demostrar o bien $\{\alpha\}^* \subseteq \mathcal{L}(M)$ o bien $\{\alpha\}^* \cap \mathcal{L}(M) = \emptyset$

Demostración: Primero, una observación. Tenemos que la única manera en la que una cadena a^k para algún k esté en el lenguaje de M es si el estado inicial es final, pues tenemos por enunciado que la transición por a siempre es desde un estado a sí mismo. Con esta observaciones, tratamos primero de probar:

$$\{a\}^* \cap \mathcal{L}(M) \neq \emptyset \iff q_0 \in F$$

Probamos los dos lados:

- $\Longrightarrow$) Que la intersección no sea vacía implica que hay una cadena a^k aceptada por el lenguaje. Supongamos que q_0 no fuese un estado final, por enunciado sabemos que por a no podemos llegar a ningún otro estado, por lo que siempre nos quedamos en q_0 pero, como este no era estado final entonces nuestro autómata en realidad no acepta la cadena a^k. Llegamos a un absurdo, que vino de suponer que q_0 no era estado final
- $\Longleftarrow$) Sabemos por enunciado que tiene una transición por a hacia sí mismo, luego, como $q_0 \in F$, tenemos que $a \in \mathcal{L}(M)$

Con esto en mente, quedaría demostrar por inducción que si acepta a la cadena a, acepta todas las cadenas a^k, y una vez probado esto, como la inclusión de q_0 en F o bien pasa (y entonces todas las cadenas $\{a\}^*$ están en el lenguaje) o bien no está (por lo que ninguna cadena está):

$$q_0 \in F \oplus q_0 \notin F \iff$$
$$\{a\} \subseteq \mathcal{L}(M) \oplus \{a\}^* \cap \mathcal{L}(M) = \emptyset \iff$$
$$\{\alpha\}^* \subseteq \mathcal{L}(M) \oplus \{a\}^* \cap \mathcal{L}(M) = \emptyset$$

(**Considerar terminar esto último, es fácil creo pero bue, lo que sí debería considerr es emprolijar esto porque es mucho mas fácil**)

□

5. Sea M = $\langle Q, \Sigma, \delta, q_0, \{q_f\}\rangle$ un AFD tal que para todos los símbolos $\alpha \in \Sigma$ se cumple que $\delta(q_0, \alpha) = \delta(q_f, \alpha)$

a) Demostrar que para todo $\omega \neq \lambda,\ \hat{\delta}(q_0, \omega) = \hat{\delta}\big(q_f, \omega\big)$

Demostración: Por inducción sobre ω:

- Caso base: $\omega = \alpha$

$\hat{\delta}(q_0, \alpha) = \delta(q_0, \alpha) = \delta\big(q_f, \alpha\big) = \hat{\delta}\big(q_f, \alpha\big)$

- Paso inductivo: $\omega = x\alpha$

$\hat{\delta}(q_0, x\alpha) = \delta\Big(\hat{\delta}(q_0, x), \alpha\Big) \underset{HI}{=} \delta\Big(\hat{\delta}\big(q_f, x\big), \alpha\Big) \underset{\text{def } \hat{\delta}}{=} \hat{\delta}\big(q_f, x\alpha\big)$ □

b) Demostrar que si ω es una cadena no vacía en $\mathcal{L}(M)$, entonces para toda k > 0, ω^k también pertenece a $\mathcal{L}(M)$

Demostración: Por inducción sobre k:

- Caso base: $k = 1$

$\omega^1 = \omega$ y por enunciado $\omega \in \mathcal{L}(M)$

- Paso inductivo: $k = n + 1$

$\omega^{n+1} \in \mathcal{L}(M) \iff \hat{\delta}\big(q_0, w^{n+1}\big) \cap F \neq \emptyset \iff \hat{\delta}\big(q_0, w^{n+1}\big) = q_f \underset{\text{Por 1)}}{\iff} \hat{\delta}\Big(\hat{\delta}\big(q_0, w^n\big), w\Big) =$

$q_f \underset{HI}{\iff} \hat{\delta}\big(q_f, w\big) = q_f$ que sabemos es cierto por a) □

6. Sea N un AFND tal que tenga a lo sumo una opción para cualquier estado y símbolo (es decir, $|\delta(q, \alpha)| = 1$), entonces el AFD D construido tiene la misma cantidad de transiciones y estados más las trancisiones necesarias para ir a un nuevo estado trampa cuando una transición no esté definida para un estado particular
7. Demostrar que para todo AFND-λ E existe un AFD D tal que $\mathcal{L}(E) = \mathcal{L}(D)$ (Usando la demo del libro)

Demostración: Sea E = $\langle Q_\lambda, \Sigma, \delta_\lambda, q_0, F_\lambda \rangle$ un AFND-λ. Definimos
D = $\langle Q_D, \Sigma, \delta_D, q_D, F_D \rangle$ de la siguiente manera:

- $Q_D = \mathcal{P}(Q_\lambda)$. En particular, va a ocurrir que todos los estados accesibles de D son subconjuntos de Q_lambda cerrados por la clausura lambda, es decir sea S el subconjunto, $S = Cl_\lambda(S)$
- $q_D = Cl_\lambda(q_0)$
- $F_D = \{S \subseteq Q_\lambda : S \cap F_\lambda \neq \emptyset\}$
- Para cada $S \subseteq Q_\lambda$ y $\alpha \in \Sigma$, definimos:
  - $\delta_D(S, \alpha) = Cl_\lambda(\{r \in Q_\lambda : \exists p \in S \wedge r \in \delta_\lambda(p, a)\})$

Ahora probamos que para una cadena $w \in \Sigma^*, w \in \mathcal{L}(E) \iff w \in \mathcal{L}(D)$:

- $\Longleftarrow$) Simplemente podemos agregar transiciones λ en todos los estados hacia el estado representando el estado trampa, y convertimos cada una de las transiciones en el equivalente de conjuntos (i.e en vez de $q_i, \{q_i\}$ en las funciones de transición)
- $\Longrightarrow$) Primero demostramos por inducción sobre la longitud de la cadena w que $\hat{\delta}_\lambda(q_0, w) = \hat{\delta}_D(q_D, w)$:
  - |w| = 0:

  $\hat{\delta}_\lambda(q_0, \lambda) = Cl_\lambda(q_0) = q_D = \hat{\delta}_D(q_D, \lambda)$
  - $|w| > 0, w = xa$:

  $\hat{\delta}_\lambda(q_0, xa) = Cl_\lambda\Big(\Big\{r \in Q_\lambda : \exists p \in \hat{\delta}_\lambda(q_0, x) \wedge r \in \delta_\lambda(p, a)\Big\}\Big) = Cl_\lambda\Big(\Big\{r \in Q_\lambda : \exists p \in \hat{\delta}_D(q_D, x) \wedge r \in \delta_\lambda(p, a)\Big\}\Big) = \hat{\delta}_D(q_D, xa)$

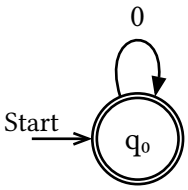
□

8. Indicar V o F y justificar
   - Si $D = \langle Q, \Sigma, \delta, q_0, F \rangle$ es un AFD entonces reconoce al menos |Q| palabras distintas, es decir $\#\mathcal{L}(D) \geq |Q|$

Falso. Como contraejemplo, un AFD que no tiene estados finales

   - Si $N = \langle Q, \Sigma, \delta, q_0, F \rangle$ es un AFND entonces todas las palabras de $\mathcal{L}(N)$ tienen longitud menor o igual a $|Q|^2$

Falso. Como contraejemplo, para $\Sigma = \{0, 1\}, L = \{w \mid 1 \notin w\}$, tenemos el AFND:



9. Cuántos AFD hay con |Q| = 2 y $|\Sigma| = 3$?

La fórmula general para la cantidad de AFDs posibles es $|Q|^{|Q| \times |\Sigma|} \times 2^{|Q|} \times |Q|$, por lo que para $|Q| = 2$ y $|\Sigma| = 3$ tenemos 512 AFDs posibles

10. Qué pasa al invertir los arcos en un AFD?

Noc si a esta pregunta le faltó algo más o quería darnos una pista sobre como arrancar a conseguir un AFD para el inverso de un lenguaje

11. Qué pasa al invertir los estados finales con no finales en un AFND?

A diferencia de con un AFD, invertir los estados no nos provee con un autómata que reconozca el complemento al lenguaje original

12. Demostrar que para cada AFND $N = \langle Q, \Sigma, \delta, q_0, F \rangle$ existe otro AFND-λ $E = \langle Q_\lambda, \Sigma, \delta_\lambda, q_0, F_\lambda \rangle$ tal que $\mathcal{L}(N) = \mathcal{L}(E)$ y F_λ tiene un solo estado final

Demostración: Definimos E de la siguiente manera:

- $Q_\lambda = Q \cup \{q_f\}$, donde q_f es un nuevo estado final
- $F_\lambda = \{q_f\}$
- $\delta_\lambda(q, \alpha) = \delta(q, \alpha)$ para todo $q \in Q, \alpha \in \Sigma$
- $\delta_\lambda(q, \lambda) = \{q_f\}$ para todo $q \in Q$ si $q \in F$ (Esto debería definirlo así, o usar la clausura lambda?)

Queremos demostrar que $\mathcal{L}(N) = \mathcal{L}(E)$, para eso primero notamos que $\hat{\delta}_N(q_0, w) = \hat{\delta}_E(q_{0\lambda}, w)$ para toda cadena $w \in \Sigma^+$. Con esto en mente tenemos que (Separamos en casos):

- $\lambda \in \mathcal{L}(N) \iff \delta(q_0, \lambda) \cap F \neq \emptyset \iff \{q_0\} \cap F \neq \emptyset \implies \delta_\lambda(q_0, \lambda) = \{q_f\} \implies \delta(q_0, \lambda) \cap F_\lambda \neq \emptyset \iff \lambda \in \mathcal{L}(E)$
- $\lambda \in \mathcal{L}(E) \iff \delta(q_0, \lambda) \cap F_\lambda \neq \emptyset \implies \delta(q_0, \lambda) \cap F \neq \emptyset \iff \lambda \in \mathcal{L}(N)$
- $w \in \mathcal{L}(N) \iff \hat{\delta}_N(q_0, w) \cap F \neq \emptyset \implies \delta_\lambda\left(\hat{\delta}_\lambda(q_0, w), \lambda\right) = \{q_f\} \iff \hat{\delta}_\lambda(q_0, w) \cap F_\lambda \neq \emptyset \iff w \in \mathcal{L}(E)$
- $w \in \mathcal{L}(E) \iff \hat{\delta}_\lambda(q_0, w) \cap F_\lambda \neq \emptyset \iff \delta_\lambda\left(\hat{\delta}_\lambda(q_0, w), \lambda\right) = \{q_f\} \implies \hat{\delta}(q_0, w) \cap F \neq \emptyset \iff w \in \mathcal{L}(N)$

□

13. Sea Σ un alfabeto con al menos dos símbolos, y sea a un símbolo de Σ. Sea $N = \langle Q, \Sigma, \delta, q_0, F\rangle$ un AFND, considerar el AFND-λ $E = \langle Q, \Sigma \setminus \{a\}, \delta_\lambda, q_0, F\rangle$ que se obtiene por reemplazar todas las transiciones por el símbolo a por trancisiones λ. Es decir:

    - Para todo $q \in Q, x \in \Sigma : x \neq a, \delta(q, x) = \delta_\lambda(q, x)$
    - Para todo $q \in Q, \delta_\lambda(q, \lambda) = \delta(q, a)$

    Determinar cuál es el lenguaje aceptado por E

14. Es posible acotar superiormente cuántas trancisiones requiere la aceptación de una palabra en un AFND-λ?

Noc (*xd*)

15. Sea $E = \langle Q, \Sigma, \delta, q_0, \{q_f\}\rangle$ un AFND-λ tal queno haya transiciones hacia q_0 ni desde q_f. Describir los lenguajes que se obtienen a partir de las siguientes modificaciones:

    a) Agregar una transición λ desde q_f hacia q_0

    b) Agregar una transición λ desde q_0 hacia cada estado alcanzables desde q_0 (notar que no es sólo aquellos directos)

    c) Agregar una transición λ hacia q_F desde cada estado que tiene un camino hacia q_f

    d) El autómata obtenido haciendo b) y c)

16. Demostrar que los lenguajes regulares son cerrados respecto de la concatenación y la clausura de Kleene mediante la construcción de un autómata.
17. Demostrar que los lenguajes regulares son cerrados respecto de la diferencia de conjuntos.
18. Demosstrar que los lenguajes regulares son cerrados respecto al reverso del lenguaje, donde el reverso está definido como el lenguaje formado por el reverso de todas sus cadenas.
19. Idem para homomorfismos y sus inversos (?
20. Sea L un lenguaje regular, y a un símbolo, llamamos L/a al cociente de L y a al conjunto de cadenas w tales que $wa \in L$. Por ejemplo, si L = {a, aab, baa} entonces L/a = {λ, ba}. Demostrar que si L es un lenguaje regular, entonces L/a también.
21. De manera similar, probar que a \ L es un lenguaje regular, donde a \ L es el conjunto de cadenas w talesque a$w \in L$
22. Demostrar que los lenguajes regulares son cerrados respecto de las siguientes operaciones:

    a) $\min(L) = \{w \mid w \in L$ y no existe α tal que $\alpha w \in L\}$

    b) $\max(L) = \{w \mid w \in L$ y no existe $\alpha \neq \lambda$ tal que $w\alpha \in L\}$

    c) init(L) = $\{w \mid$ Para algún $x, wx \in L\}$
23. La mayoría de los ejs de la sección 4.2 (Me dió fiaca seguir copiando xd)
24. Sea L un lenguaje regular, y sea n la constante del Lema de Pumping para L. Determinar veracidad y demostrar:

    a) Para cada cadena z en L, con $|z| \geq n$, la descomposición de z en uvw, con $|v| \geq 1$ y $|uv| \leq n$, es única.

    b) Cada cadena z en L, con $|z| \geq n$, es de la forma uv^iw para algún u, v, w con $|v| \geq 1$ y $|uv| \leq n$ y algún i

    c) Hay lenguajes no regulares que satisfacen la condición afirmada por el Lema de Pumping

    d) Sean L_1, L_2 lenguajes sobre el alfabeto Σ tal que $L_1 \cup L_2$ es un lenguaje regular. Entonces L_1 y L_2 son regulares.
25. Sea $\mathcal{C}$ el mínimo conjunto que contiene todos los lenguajes finitos, y está cerrado por unión finita, intersección, complemento, y concatenación ¿Cuál es la relación entre $\mathcal{C}$ y el conjunto de todos los lenguajes regulares?
26. Dar un algoritmo de desición que determine si el lenguaje aceptado por un autómata finito es el conjunto de todas las cadenas del alfabeto
27. Dar un algoritmo de decisión que determine si el lenguaje aceptado por un autómata finito es cofinito
28. Demostrar que todo lenguaje regular es libre de contexto. **Pista**: construir una gramática mediante inducción en la cantidad de operadores de una expresión regular
29. Una GLC es linear a derecha si el cuerpo de cada producción tiene a lo sumo una variable, y la misma aparece más a la derecha. Es decir, todas las producciones son de la forma $A \to wB$ o $A \to w$.

a) demostrar que toda GLC lineal a derecha genera un lenguaje regular. **Pista**: construir un autómata finito λ que simule la derivación más a izquierda de la gramática, con sus estados representando el símbolo no terminal de la forma sentencial actual.

b) Demostrar que todo lenguaje regular tiene una GLC lineal a derecha. **Pista**: Empezar un AFD y hacer que las variables de la gramática representen estados

30. Considerar la gramática G definida por las producciones:

$$S \to aS \mid Sb \mid a \mid b$$

a) Demostrar por inducción sobre la longitud de la cadena que ninguna cadena generada por G tiene ba como subcadena.

b) Identificar $\mathcal{L}(G)$

31. Sea G la gramática con producciones:

$$S \to aSbS \mid bSaS \mid \lambda$$

Demosstrar que $\mathcal{L}(G)$ es es conjunto de cadenas con una misma cantidad de $a$es que de bs

32. Supongamos que G es una GLC sin producciones que tengan a λ del lado derecho. Si w está en el lenguaje de G, $|w| = n$ y $S \overset{m}{\Rightarrow} w$, demostrar que w tiene un arbol de derivación con n + m nodos.

33. Supongamos lo mismo que en el ej anterior, pero ahora puede haber producciones con λ en la derecha. Demostrar que un árbol de derivación para w puede tener a lo sumo $n + 2m - 1$ nodos.

34. Demostrar que si $X_1X_2...X_k \overset{*}{\Rightarrow} a$ entonces todos los símbolos que provienen de derivar X_i están a la izquierda de todos los que provienen de derivar X_j, si $i < j$. **Pista**: Usar inducción sobre la cantidad de pasos en la derivación

35. Indicar veracidad:
    - Si $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ es un autómata de pila entonces cada cadena $w \in \mathcal{L}_\lambda(P)$ es reconocida por P en a lo sumo $|w| * \#Q * \#\Gamma$ transiciones, es decir:

$$\text{Sea } n \leq |w| * \#Q * \#\Gamma, \text{ entonces existe } p \in Q \text{ tal que } (q_0, w, Z_0) \overset{n}{\vdash} (p, \lambda, \lambda)$$

36. Dado un autómata finito $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, dar un autómata de pila $M' = \langle Q', \Sigma, \delta', q_{0'}, Z_0', \emptyset\rangle$ tal que $\mathcal{L}(M) = \mathcal{L}_\lambda(M')$.

37. Consideremos la demostración del teorema que afirma que para cada autómata $M = \langle Q, \Sigma, \delta, q_0, Z_0, F\rangle$ existe un autómata M' tal que $\mathcal{L}(M) = \mathcal{L}_\lambda(M')$.
¿Si M es determinístico, entonces el autómata M' construido en la demostración también lo es?

38. Consideremos la demostración del teorema que afirma que dado $M' = \langle Q', \Sigma, \delta', q_{0'}, X_0, \emptyset\rangle$ existe un autómata M tal que $\mathcal{L}_\lambda(M') = \mathcal{L}(M)$.
¿Si M' es determinístico, entonces el autómata M construido en la demostración también lo es?

39. Demostrar que si P es un APDm entonces existe un APD P_2 con solo dos símbolos de pila (es decir $|\Gamma_2| = 2$) tal que $\mathcal{L}_\lambda(P) = \mathcal{L}_\lambda(P_2)$. **Pista**: Considerar una codificación binaria para la pila

40. Un APD está restringido si en toda transición puede incrementar la altura de la pila con a lo sumo un símbolo, es decir, para toda transición $\delta(q, w, Z)$ que contiene algún (p, γ), debe ocurrir que $|\gamma| \leq 2$. Demostrar que si P es un APD, entonces existe un APD restringido P_2 tal que $\mathcal{L}(P) = \mathcal{L}(P_2)$

41. Demostrar que si P es un APD, entonces existe un APD P_1 de un solo estado tal que $\mathcal{L}_\lambda(P) = \mathcal{L}_\lambda(P_1)$
42. Suponiendo que P es un APD tal que tiene s estados, t símbolos de pila, y ninguna regla en la cual lo que se apila tenga longitud mayor a u, dar una cota ajustada para la cantidad de variables en la GLC construida según el metodo de la demo.
43. Existe, para todo lenguaje libre de contexto sin λ, una gramática tal que todas sus producciones son de la forma $A \to BCD$ (un cuerpo con sólo tres variables) o $A \to a$ (cuerpo con sólo un terminal)? Demostrar o dar contraejemplo (**Si no usaaste la sección de FNC salteate este**)
44. Hacer el ej 7.1.11 del libro de Hopcraft
45. Sea L un lenguaje. Si todas las cadenas de L validan el lema de pumping para lenguajes libres de contexto, ¿Se puede concluir que L es libre de contexto?
46. Sea L un lenguaje regular. Demostrar que todas las palabras de L validan el Lema de Pumping para lenguajes libres de contexto
47. Mostrar que $L = \{a^p : p \text{ es número primo}\}$ no es libre de contexto. **Pista**: Asumir L libre de contexto, y sea n la longitud dada por el lema de pumping. Sea m el primo mayor o igual a n, considerar $\alpha = a^m$ y bombear m+1 veces
48. Demostrar que hay lenguajes que pueden ser reconocidos con un autómata de dos pilas pero no por uno con sólo una pila.
49. Demostrar que los lenguajes libres de contexto son cerrados respecto de las siguientes operaciones:
    - Ini(L). **Pista:** Comenzar con una FNC para L
    - L / a. **Misma que antes**
50. Completar las demostraciones de propiedad de clausuras para lenguajes libres de contexto
51. Eks 7.3.4 y 7.3.5 del libro
52. Dar algoritmos para decidir si:

    a) L es finito b) L contiene al menos 100 cadenas c) Dada una GLC G y una variable A, decidir si A es el primer símbolo en alguna forma sentencial
53. Demostrar que para cualquier GLC, todos los árboles de derivación oara cadenas de longitud n tienen $2n - 1$ nodos internos