>>> IF013 - Fundamentos Teóricos de Informática
>>> Licenciatura de Sistemas - UNPSJB - Sede Trelew

Name: Celia Cintas[†], Pablo Navarro[‡], Samuel Almonacid[§]
Date: August 14, 2017

[†]cintas@cenpat-conicet.gob.ar, cintas.celia@gmail.com, @RTFMCelia
[‡]pnavarro@cenpat-conicet.gob.ar, pablo1n7@gmail.com
[§]almonacid@cenpat-conicet.gob.ar, almonacid.samuel.tw@gmail.com

>>> Unidad 1

1. Autómatas finitos. Reconocedores. Traductores. Diagrama de estados. Autómatas finitos no deterministas.
2. Equivalencia entre autómatas finitos deterministas y no deterministas. Morfismos sobre autómatas. Autómata Cociente.
3. Propiedades de lenguajes aceptados por Autómatas Finitos. Expresiones y lenguajes regulares.
4. Propiedades algebraicas de los lenguajes regulares. Equivalencia entre autómatas finitos y lenguajes regulares.
5. Teorema de Kleene. Gramáticas regulares. Relación entre gramáticas regulares y autómatas finitos.
6. Usos y aplicaciones de los autómatas finitos y lenguajes regulares.

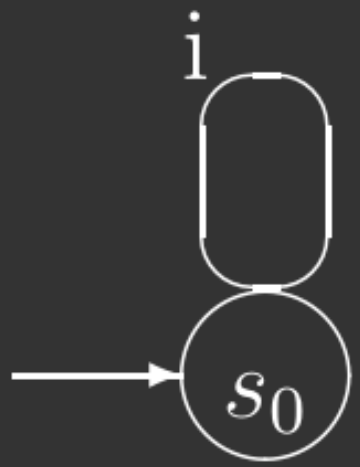
Teorema

Un lenguaje L es regular si y sólo si es reconocido por un AF.

Demostración

La clase de lenguajes regulares es la clase más chica de lenguajes que contiene al conjunto $\emptyset$, a λ, a los símbolos del alfabeto y es cerrada bajo unión, concatenación y estrella de Kleene. La prueba será por construcción del AF.

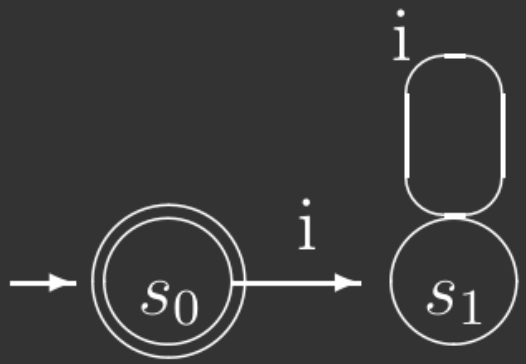
* El menor AF para $L = \emptyset$ es $M = (S, \Sigma, \delta, s_0, F)$, $S = \{s_0\}$, $i \in \Sigma$, $F = \emptyset$.



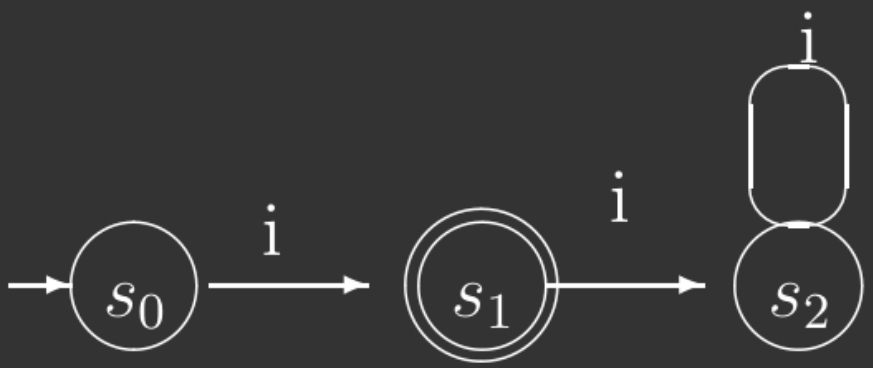
>>> Teorema de Kleene (Cont.)

Demostración

* El menor autómata finito que reconoce λ es
  $M = (S, \Sigma, \delta, s_0, F),\ S = \{s_0, s_1\},\ i \in \Sigma,\ F = \{s_0\}.$



* El menor autómata finito que reconoce i es
  $M = (S, \Sigma, \delta, s_0, F),\ S = \{s_0, s_1, s_2\},\ i \in \Sigma,\ F = \{s_1\}.$



Demostración

* Queremos probar que $L_1 \cup L_2$ es un lenguaje aceptado por un AF. Lo demostraremos por construcción:
  Sean $M_1 = (S_1, \Sigma, \delta_1, s_1, F_1)$ y $M_2 = (S_2, \Sigma, \delta_2, s_2, F_2)$.
  Construimos un autómata finito M no determinista que acepte $L(M_1) \cup L(M_2)$.
  $M = (S, \Sigma \cup \{\lambda\}, \delta, s, F)$, donde:
  * $S = S_1 \cup S_2 \cup \{s\}$ (s es el estado inicial).
  * $F = F_1 \cup F_2$.
  * $\delta = \delta_1 \cup \delta_2 \cup \{\delta(s, \lambda) = \{s_1, s_2\}\}$.

  Por lo que: $\delta(s, w) \supseteq q, q \in F$, si y solo si:
  $\delta_1(s_1, w) \supseteq q, q \in F_1$ o $\delta_2(s_2, w) \supseteq q, q \in F_2$. Por lo tanto $L(M) = L(M_1) \cup L(M_2)$.

Demostración

* La concatenación ($L(M) = L(M_1) \cdot L(M_2)$) se demuestra de manera simil al punto anterior.
* Solo queda demostrar que $L(M) = L(M_1)^*$, donde M_1 es un AF y M es un AFND que construiremos. dado $M_1 = (S_1, \Sigma, \delta_1, s_1, F_1)$ construimos M tal que tiene todos los estados de M_1 y un nuevo estado inicial s_1 . Este nuevo estado también es aceptador, para poder reconocer λ.
  $S = S_1 \cup \{s_1\}$
  s es el estado inicial, $s \notin S_1$
  $F = F_1 \cup \{s\}$
  $\delta = \delta_1 \cup \{\delta(s, \lambda) = s_1\} \cup \{\delta(s_i, \lambda) = s_1, s_i \in F_1\}$

>>> Propiedades de los Lenguajes Regulares

Ahora veremos clausura bajo ciertas operaciones, que nos permitirán una mayor comprensión del concepto de lenguajes regulares.

* brindan herramientas para la construcción y simulación de AF.
* esclarecen aún mas el nexo entre los autómatas finitos y las expresiones regulares.
* muestra que los LR son estables.
* ayudan a identificar el tipo de un lenguaje.

Definición

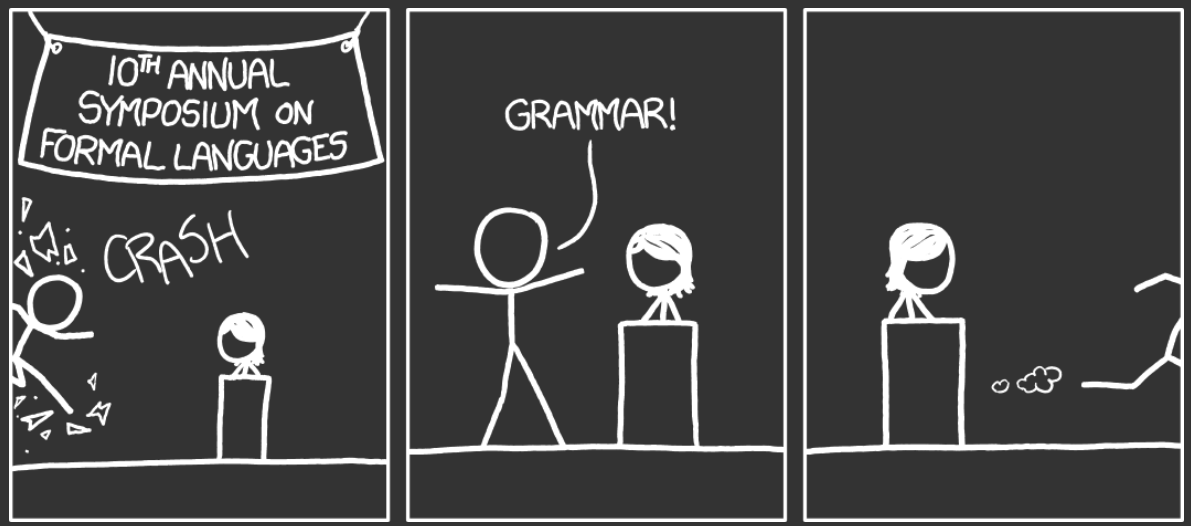
La clase de los lenguajes aceptados por AF, es decir la clase de los lenguajes regulares, es cerrada bajo: unión, concatenación, estrella de Kleene, complemento e Intersección.

Problemas decidibles sobre LR:

* Pertenencia: dado un lenguaje regular L y $\alpha, \alpha \in \Sigma^*$, α pertenece a L?
* Finitud: dado L, es L finito?
* Vacuidad: es L vacío?
* Equivalencia: dados L_1 y L_2 son equivalentes?

>>> Gramáticas

Estudiaremos uno de los tipos de generadores de lenguajes formales: las **gramáticas estructuradas por frases**. Estos dispositivos comienzan a partir de un iniciador designado con antelación y su operación está limitada por un conjunto de reglas. La teoría que describe los generadores de lenguajes, es decir las gramáticas, complementa a la de autómatas. Ambos son necesarios en la especificación y análisis de los lenguajes de computación.



Definición

Definiremos gramática estructurada por frases (GEF), como una cuádrupla (V_n, V_t, S, P) donde:

* V_n es un conjunto finito de símbolos no terminales o símbolos auxiliares.
* V_t es un conjunto finito de símbolos terminales.
* S es el símbolo inicial.
* P conjunto finito de reglas de producción de la forma $\alpha \to \beta$ donde $\alpha \in (V_n \cup V_t)^+ V_n (V_n \cup V_t)^+$ y $\beta \in (V_n \cup V_t)^*$. Las reglas de producción nos permiten generar las palabras.

De aquí en adelante, asumiremos que $V_n \neq \emptyset, V_t \neq \emptyset \; y \; V_n \cap V_t = \emptyset$

$$G = (V_n, V_t, S, P)$$
$$V_n = \{S\}$$
$$V_t = \{0, 1\}$$
$$P = \{S \to 0S, S \to 1\}$$

Definición

Sea $G = (V_n, V_t, S, P)$ una gramática y $\phi, \mu \in (V_t \cup V_n)$, diremos que $\mu\alpha\phi$ deriva directamente a $\mu\beta\phi$ en G, y lo escribiremos como $\mu\alpha\phi \xrightarrow{G} \mu\beta\phi$ si existe una producción $\alpha \to \beta$ en P.

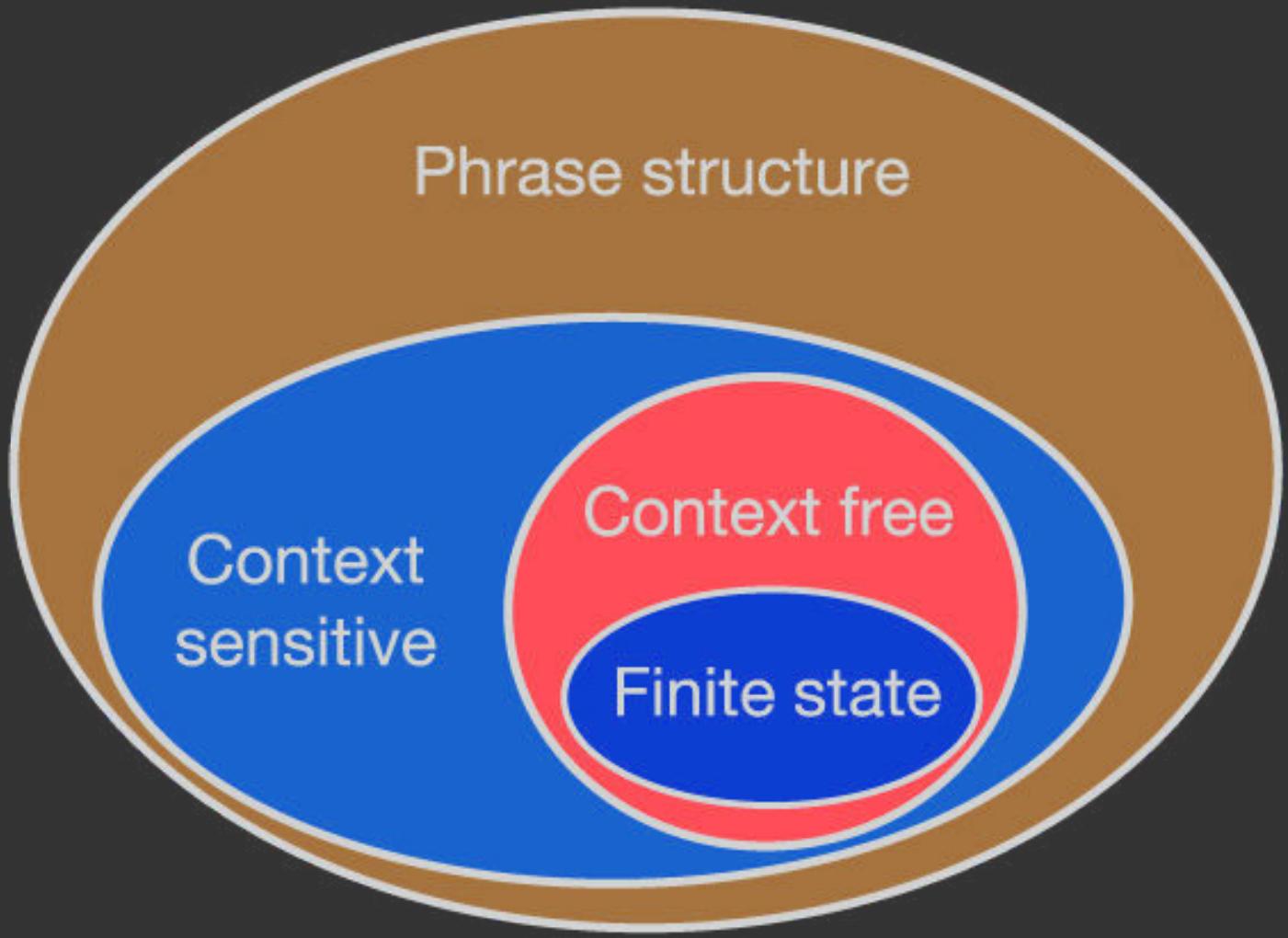
Definición

Dadas dos cadenas α, β se dice que α deriva a β según la gramática G , notado $\alpha \xrightarrow{G+} \beta$, si β puede obtenerse por aplicación de reglas de producción de G a partir de α, es decir: $\alpha \xrightarrow{G} \gamma_1, \xrightarrow{G} \gamma_2, \cdots, \gamma_{n-1} \xrightarrow{G} \gamma_n, \gamma_n \xrightarrow{G} \beta$

Definición

Sea una gramatica $G = (V_n, V_t, S, P)$, $L(G) = \{w \in V_t^* | S \xrightarrow{G+} w\}$ es el lenguaje generado por G.

>>> Clasificación de Chomsky



Computational and volutionary aspects of language. Martin A. Nowak, Natalia L. Komarova and Partha Niyogi, 2002. Nature.

Definición

Una gramática regular (GR), como una cuádrupla (V_n, V_t, S, P) donde:

* V_n es un conjunto de símbolos no terminales.
* V_t es un conjunto de símbolos terminales.
* S es el símbolo inicial.
* P conjunto finito de reglas de producción de la forma $\alpha \to \beta$ tales que:
  * α es un solo no terminal, $\alpha \in V_n$.
  * β es un solo terminal o es un terminal concatenado con un no terminal, es decirm $\beta = a$ o $\beta = aB$, donde $a \in V_t$ y $B \in V_n$

>>> Gramáticas Regulares (Tipo 3) (Cont.)

Definición

Si todas las producciones son de la forma $A \to xB$ o $A \to x$, donde $A, B \in V_n$ y $x \in V_t$. Entonces la gramática es llamada **lineal a derecha**.

Definición

Si todas las producciones son de la forma $A \to Bx$ o $A \to x$, donde $A, B \in V_n$ y $x \in V_t$. Entonces la gramática es llamada **lineal a izquierda**.

$$G = (V_n, V_t, S, P)$$
$$V_n = \{S, B\}$$
$$V_t = \{a, b\}$$
$$P = \{S \to a, S \to aB, B \to bB, B \to aB, B \to b, B \to a\}$$

Cómo sabemos si la cadena $abbaba \in L(G)$??

$S \to aB \to abB \to abbB \to abbaB \to abbabB \to abbaba$

Teorema

Sea G una gramática regular. $L(G)$ es un lenguaje generado por G si y solo si existe un autómata finito M que reconoce $L(M)$, tal que $L(M) = L(G)$.

Demostración

Veamos la demostración en dos partes:

* Sea $G = (V_n, V_t, S, P)$ una GR, existe un AF M tal que si $x \in L(M)$ entonces $x \in L(G)$. Consideremos a $M = (K, V_t, \delta, S, F)$. Dada G, espicificaremos el resto del AFND, M de la siguiente manera:
  1. Los estados de M son V_n de G más un estado adicional A. $K = V_n \cup \{A, R\}$; $A, R \notin V_n$
  2. $F = \{A\}$, excepto que P tenga $S \to \lambda$, en ese caso $F = \{A, S\}$.

Demostración

* Definimos δ considerando los siguientes casos:
  * $\delta(B, a) \supseteq \{A\}$, $si\ 'B \to a' \in P; a \in V_t; B, A \in V_n$.
  * $\delta(B, a) \supseteq \{C\}$, $si\ 'B \to aC' \in P;\ a \in V_t;\ B, C \in V_n$.
  * $\delta(B, a) \supseteq \{R\}, R \notin F$, para todo par $(B, a) \in S \times \Sigma$ no considerado en la definición del autómata.

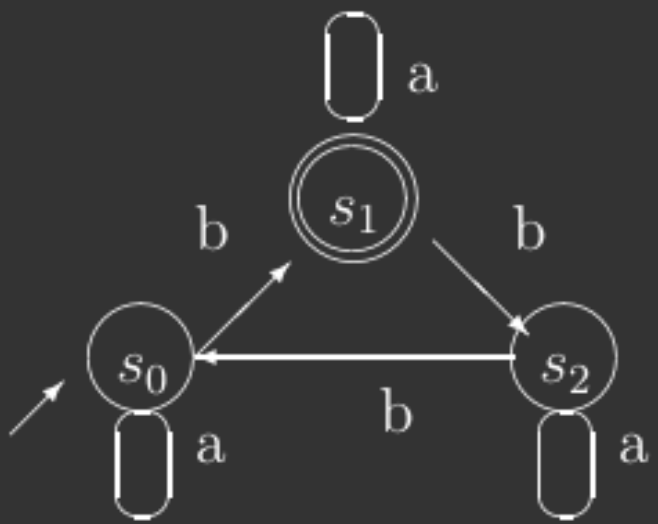
Ahora consideremos la segunda parte: Dado un autómata finito M existe una GR G tal que si $x \in L(G)$ entonces $x \in M(G)$. A partir de un AFND $M = (S, \Sigma, \delta, s_0, F)$ Definamos una gramática $G = (S, \Sigma, s_0, P)$ donde P esta formada por:

* $B \to aC$, si $\delta(B, a) = C$.
* $B \to a$, si $\delta(B, a) = C$ y $C \in F$.

De esta manera definimos un AFND a partir de una GR. Ahora solo queda en chequear que la cadena x se puede generar con G y que x es cadena aceptada en el AF M.

$$M = (S, \Sigma, \delta, s_0, F)$$
$$S = \{s_0, s_1, s_2\}$$
$$\Sigma = \{a, b\}$$
$$F = s_1$$

δ	a	b
s_0	s_0	s_1
s_1	s_1	s_2
s_2	s_2	s_0



$$G = (V_n, V_t, S_0, P)$$
$$V_n = \{S_0, S_1, S_2\}$$
$$V_t = \{a, b\}$$

$$P = \{S_0 \to aS_0, S_0 \to bS_1\,,$$
$$S_0 \to b, S_1 \to aS_1,$$
$$S_1 \to bS_2, S_1 \to a,$$
$$S_2 \to aS_2, S_2 \to bS_0\}$$

>>> Conversión de Gramáticas Regulares por Derecha a Izquierda

Definición

Para cada Gramática Lineal por Derecha existe una Gramática Lineal por Izquierda que genera el mismo lenguaje y viceversa.

Los pasos a seguir son:

- Se transforma la gramática de forma que ninguna regla tenga el *Start* del lado derecho.
  - Se crea un nuevo $S', S' \in V_n$.
  - Se crea una nueva producción $\forall S \to x$ se tiene $S' \to x$
  - Cada regla $A \to xS$ se transforma a $A \to xS'$.

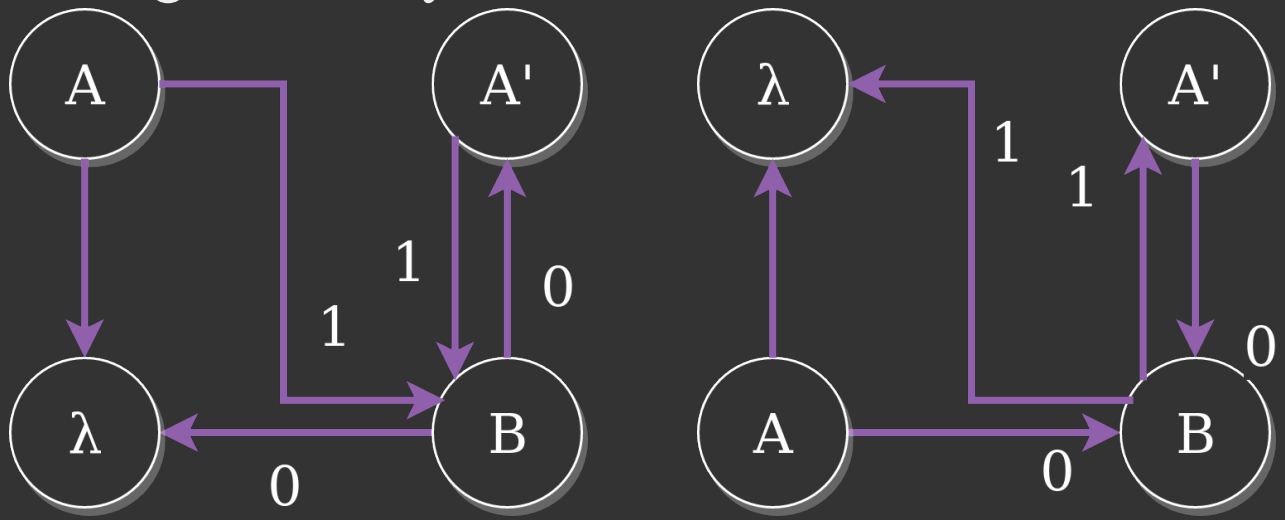
>>> Conversión de Gramáticas Regulares por Derecha a Izquierda (Cont.)

* Se crea un grafo dirigido G.
  * $\forall A \in V_n \cup \{\lambda\}$ se crea un nodo.
  * $\forall A \to aB \in P$ se crea un arco etiquetado.
  * $\forall A \to a \in P$ se crea un arco etiquetado direccionado a λ.
  * Si $\exists S \to \lambda$ se crea un arco al nodo λ sin etiquetar.
* Se crea otro grafo G' a partir de G.
  * Se intercambian las etiquetas del *Start* y λ.
  * Se invierten la dirección de todos los arcos.
* Se transforma en un conjunto de reglas:
  * $\forall$ nodo, se crea un símbolo no terminal excepto para el nodo λ.
  * $\forall$ arco etiquetado con a que va del nodo A a B, $B \in V_n \cup \lambda$ se crea $A \to Ba$
  * Si $\exists$ un arco del nodo *Start* al nodo λ se crea una regla $S \to \lambda$.

>>> Conversión de Gramáticas Regulares por Derecha a Izquierda (Cont.)

Sea $G_1 = V_t = \{0, 1\}, V_n = \{A, B\}, P : \{A \to 1B|\lambda, B \to 0A, B \to 0\}$

1. Creamos $A' \to 1B|\lambda$.
2. Creamos $B \to 0A'$.
3. Eliminamos $A' \to \lambda$.
4. Creamos grafo G y G':

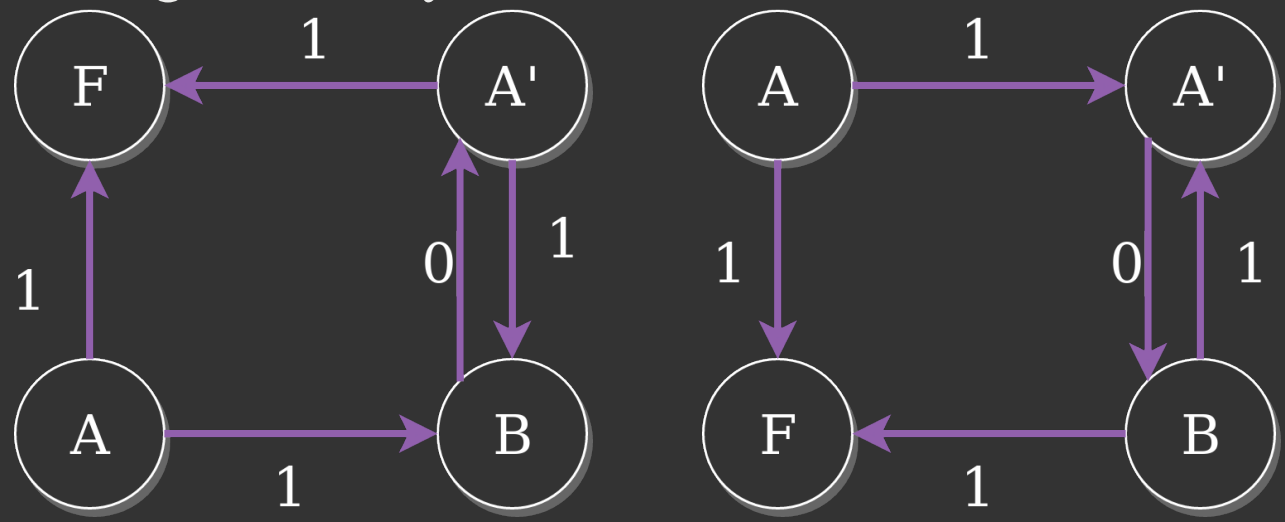


$P' = \{A \to B0, A \to \lambda, B \to A'1, B \to 1, A' \to B0\}$

>>> Conversión de Gramáticas Regulares por Izquierda a Derecha (Cont.)

Sea $G_1 = V_t = \{0, 1\}, V_n = \{A, B\}, P : \{A \to B1|1, B \to A0\}$

1. Creamos $A' \to B1|1$.
2. Creamos $B \to A'0$.
3. Creamos grafo G y G':



$P' = \{A \to 1A', A \to 1, B \to 1A', B \to 1, A' \to 0B\}$

Sea $P = \{S \to aaS|bS|abA|bA|b, A \to aA|bba\}$

La ER se verá cómo:

$S = (aa + b)S + (ab + b)A + b$
$A = aA + bba = a^*bba$

Sustituimos A en S:

$S = (aa + b)^*((ab + b)a^*bba + b)$ $ER = (aa + b)^*((ab + b)a^*bba + b)$

>>> Pumping Lemma

Hasta ahora hemos visto diversas formas de mostrar que un lenguaje es regular, pero ninguna (aparte de que no nos funcione nada de lo que sabemos hacer) para mostrar que no lo es. Veremos ahora una herramienta para demostrar que un cierto L no es regular.

Teorema

Sea L un lenguaje regular. Entonces existe un número $N > 0$ tal que toda cadena $w \in L$ de largo $|w| > N$ se puede escribir como $w = xyz$ de modo que $y \neq \lambda$, $|xy| \leq N$, y $\forall n \geq 0, xy^n z \in L$.

Demostración

Sea $M = (K, \Sigma, \delta, s, F)$ un AFD que reconoce L. Definiremos $N = |K|$ Al leer w, M consume los primeros N caracteres de w, sea q_i al estado que se llega luego de consumir $w_1 w_2 \cdots w_i$: $(q_0, w_1 w_2 ..) \vdash (q_0, w_2 ..) \vdash \cdots (q_i, w_{i+1} ..) \vdash (q_N, w_{N+1} ..) \vdash \cdots$
Los estados $q_0, q_1, \cdots, q_N$ no pueden ser todos distintos por $|K| = N$. Por lo que algún camino se repite y si lo eliminamos de w, M llegará a un estado aceptador. De la misma manera podemos duplicar y.

>>> Pumping Lemma (Cont.)

¿Cómo utilizar el Lema de Bombeo para demostrar que un lenguaje no es regular? La idea es negar las condiciones del Teorema anterior.

1. Para cualquier longitud N,
2. debemos ser capaces de elegir alguna $w \in L$, $|w| > N$,
3. de modo que para cualquier forma de partir $w = xyz$, $y \neq \lambda, |xy| \leq N$,
4. podamos encontrar alguna $n \geq 0$ tal que $xy^n z \notin L$.

Dado $L = \{a^n b^n | n > 0\}$. Probar que no es regular. Supongamos que L es regular; sea k el entero del que habla el teorema. Tomemos la cadena $w = a^k b^k \in L$ (en este caso, $long(a^k b^k) = 2k > k$) y veamos que no es posible descomponer la cadena w en la forma xyz, tal que $xy^i z \in L$, para todo $i \geq 0$.

Caso 1 tomamos y enteramente formada por letras a:

$x = a^p$

$y = a^q$

$z = a^r b^s$ donde $s = p + q + r$ siendo $p, r \geq 0$ y $q, s > 0$.

Por lo tanto $xy^n z = a^{(p+nq+r)} b^s$ para cada $n \geq 0$, pero si $s = p + q + r$ luego $s \neq p + nq + r$ salvo para $n = 1$, luego $xy^n z \in L$ si $n = 1$

Caso 2 y contiene todas b:

$x = a^p b^r$

$y = b^q$

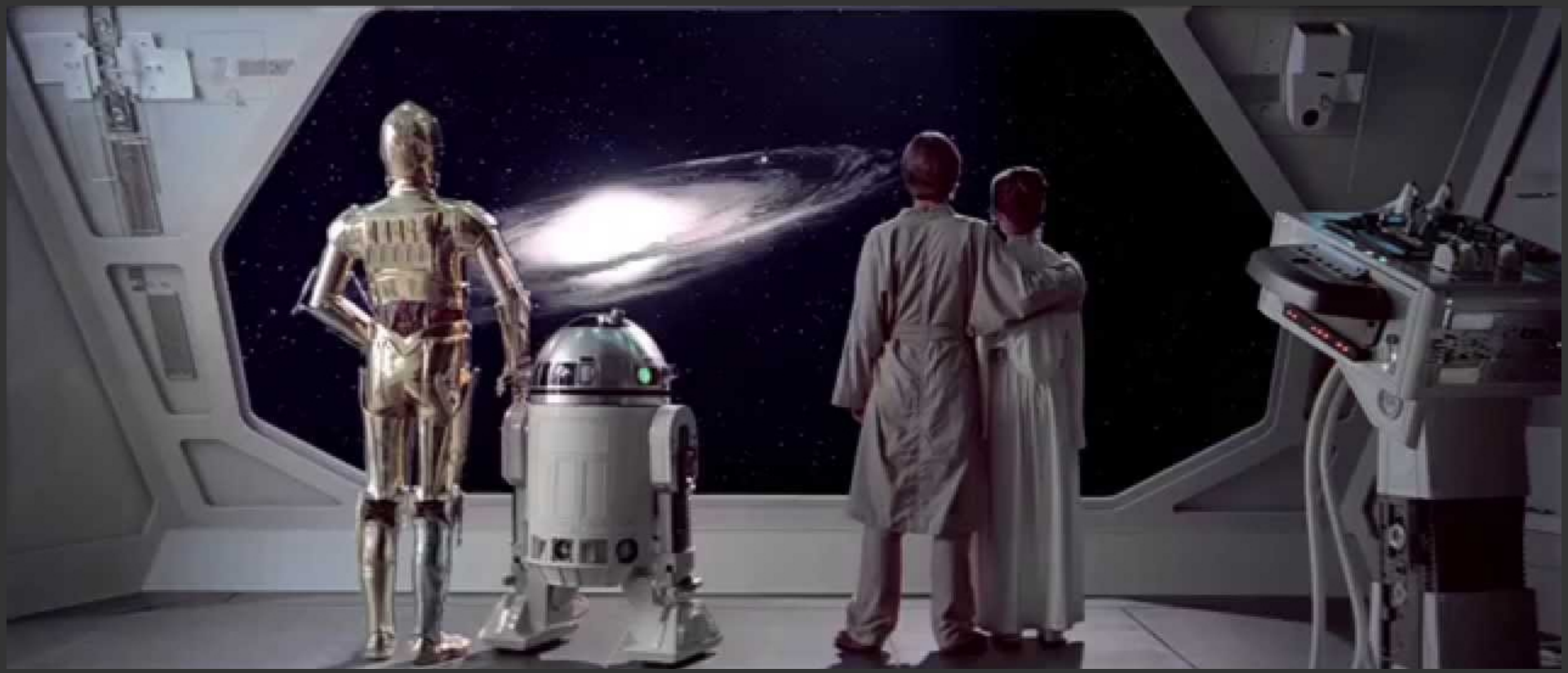
$z = b^s$ donde $q > 0, s \geq 0$ y $s + q + r = p$

Pero si $xy^n z = a^p b^{(r+nq+s)}$ debería ser $s + nq + r = p$, pero nuevamente, esto vale sólo cuando $n = 1$

Caso 3 y contiene a y b, entonces en $xy^n z$ hay ocurrencias de b que preceden las letras a y la cadena no mantiene la forma.

Por lo tanto, no hay forma de descomponer $w = xyz$ tal que $xy^n z$ pertenezca al lenguaje para todo valor de $n \geq 0$. Esto implica que L no es regular.

>>> Gracias!



Bibliografía

1. Introduction to Automata Theory, Languages, and Computation - Hopcroft et. al 2007 (3er ed.)
2. Teoría de la Computación - Gonzalo Navarro 2011.
3. Fundamentos de Cs. de la Computación - Juan Carlos Augusto 1995.