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

Name: Celia Cintas[†], Pablo Navarro[‡], Samuel Almonacid[§]
Date: August 9, 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.

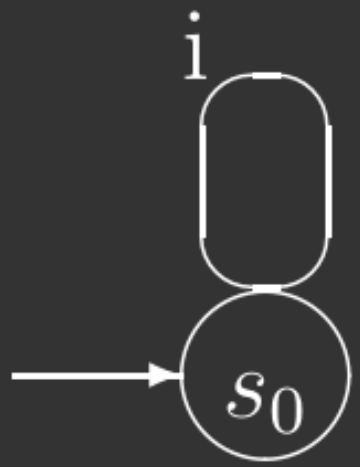
Teorema

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

Demostración

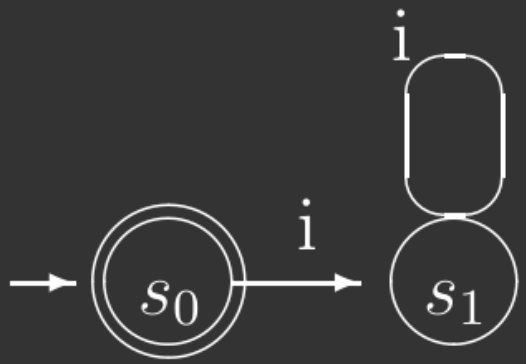
La clase de lenguajes regulares es la clase más chica de lenguajes que contiene al conjunto $\emptyset$, a λ, a los símbolos del alfabeto y es cerrada bajo unión, concatenación y estrella de Kleene. La prueba será por construcción del AF.

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

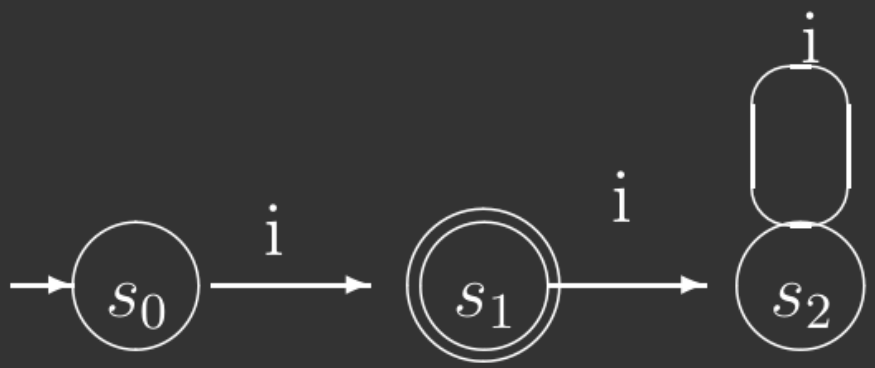


Demostración

* 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

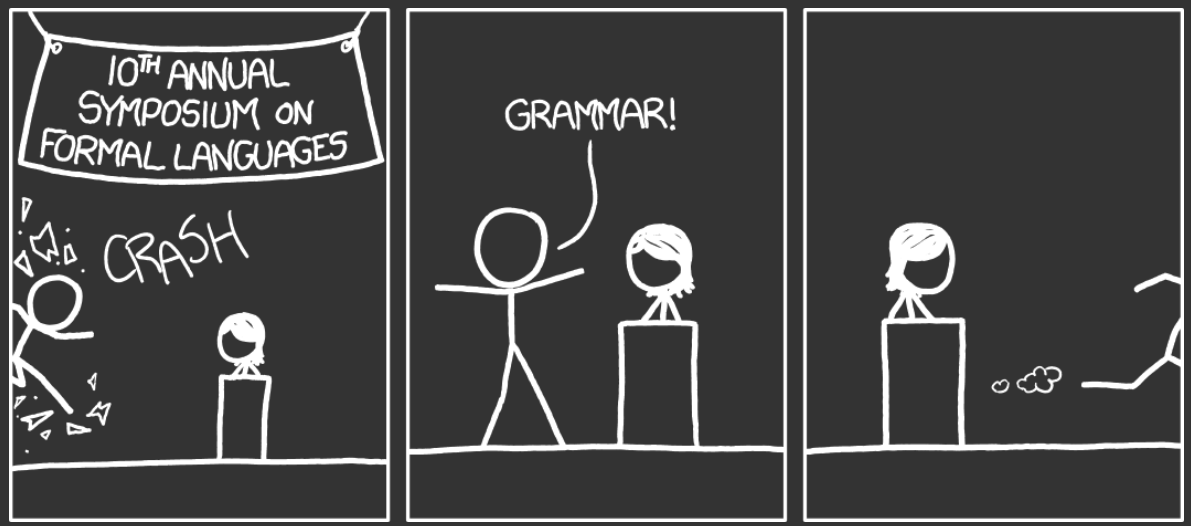
La clase de los lenguajes aceptados por AF, es decir la clase de los lenguajes regulares, es cerrada bajo: unión, concatenación, estrella de Kleene, complemento e Intersección.

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.

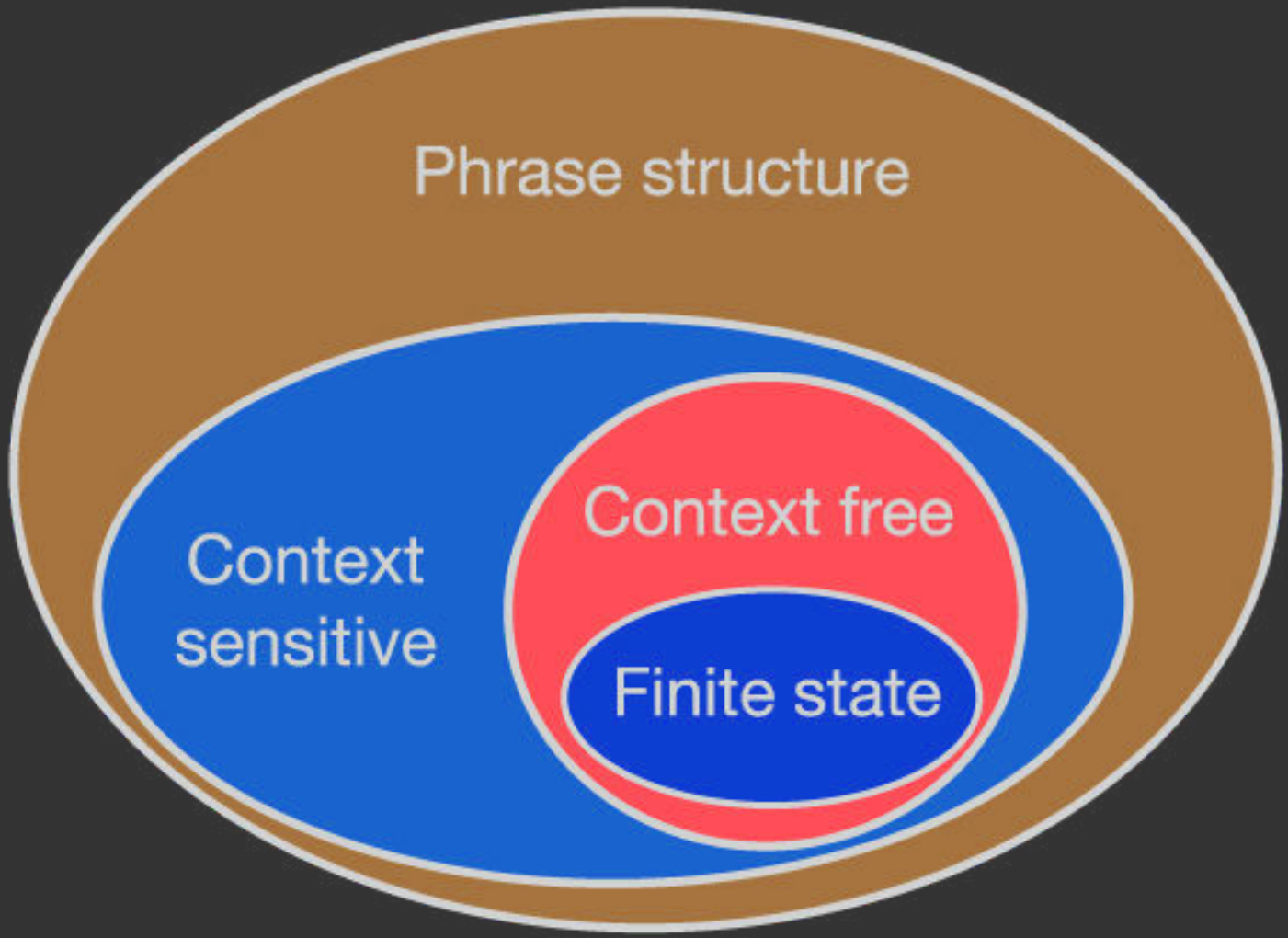
Definición

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.

>>> Gramáticas Regulares (Tipo 3)

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.

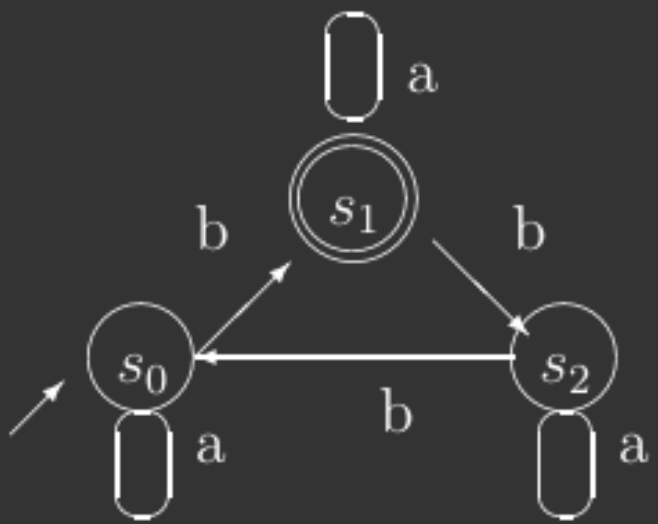
Ahora consideremos la segunda parte: Dado un autómata finito M existe una GR G tal que si $x \in L(G)$ entonces $x \in M(G)$. A partir de un AFND $M = (S, \Sigma, \delta, s_0, F)$ Definamos una gramática $G = (S, \Sigma, s_0, P)$ donde P esta formada por:

* $B \to aC$, 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



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

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

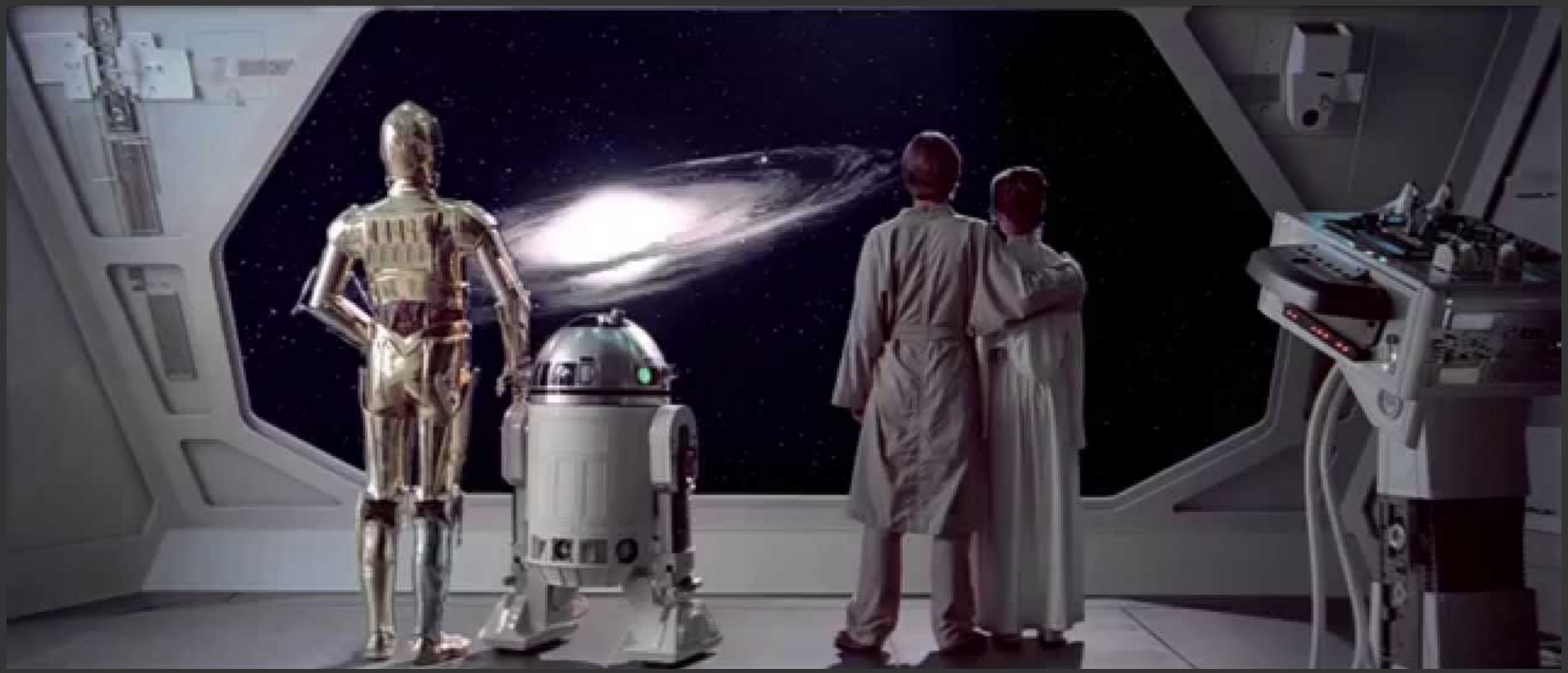
>>> Gramáticas Regulares y Expresiones Regulares

something here

>>> Pumping Lemma para AFD

something here

>>> 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.