Autómatas

Composición de autómatas

- k: Conjunto no vacío de estados
- Σ: Alfabeto de símbolos (entrada)
- $s \in K$: estado inicial
- $F \subseteq K$: Conjunto de estados finales
- Γ: Alfabeto de símbolos (pila)
- $\delta : K \times \Sigma \rightarrow K$: (AFD) **Función** de transición. Al ser determinista, transitar es función
- Δ: $K \times \Sigma^* \times K$: (AFND) Relación de transición
- Δ: $(K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$: (APND) Relación de transición

Transitar

- **Directamente** : Pasar de una configuración a la siguiente. Relación binaria
- **En N pasos (transitar a secas**): Secuencia finita transiciones. Cierre reflexivo y transitivo
- **En al menos 1 paso**: Transitar en N>1 pasos. Cierre transitivo

Computación

Secuencia finita de configuraciones en las cuales $C_{n-1} \vdash C_n$. La longitud de una computación es el número de transiciones requeridas para ir de una configuración a otra. Además:

- **Computación bien iniciada**: Cualquier computación donde partimos de la configuración inicial
- **Computación bien terminada**: Cualquier computación donde ultima configuración es una configuración terminal
- **Computación completa**: Es bien iniciada y bien terminada a la vez
- **Computación bloqueada (ND)**: Si su última configuración es de bloqueo

Otras definiciones

- **Estado terminal**: Último estado de la última configuración de una computación terminada
- **Configuración de bloqueo (ND)**: Cuando no puede transitar y no es configuración final
- **Cadena aceptada**: Un autómata acepta una cadena cuando existe una computación completa sobre esa cadena $\exists q \in F \mid (s, w) \vdash^* (q, \varepsilon)$
- **Lenguaje aceptado por M**: Conjunto de todas las cadenas aceptadas por M
- **Diagrama de estados**: Grafo dirigido y etiquetado donde los nodos son los estados. Los estados finales se marcan con doble círculo y el estado inicial se marca con '>'
- **Estado accesible**: Un estado q es accesible sii $\exists x \in \Sigma^* \mid (s, x) \vdash^* (q, \varepsilon)$
- **Estado inaccesible**: Sii q no es accesible
- **Autómatas finitos equivalentes** $(M_1 \equiv M_2)$: Dos autómatas son equivalentes sii $L(M_1) = L(M_2)$

Tabla de propiedades

	AFD	AFND	APND
Expresión	$(k, \Sigma, \delta, s, F)$	$(k, \Sigma, \Delta, s, F)$	$(K, \Sigma, \Gamma, \Delta, s, F)$
Configuración	$(q, w) \in K \times \Sigma^*$	$(q, w) \in K \times \Sigma^*$	$(q, w, p) \in K \times \Sigma^* \times \Gamma^*$
Configuración inicial	(s, w)	(s, w)	(s, w, ε)
Configuración terminal	(q, ε)	(q, ε)	$(q, \varepsilon, \varepsilon)$
Configuración de bloqueo	NO	SI	SI
Transitar directamente	$(q, \sigma w' \vdash (\delta(q, \sigma), w')$	$((q, uw'), (q', w'))$	$((q, w, p), (q', w', p'))$
Lenguaje generado	$L.3$	$L.3$	$L.2$

AFD

- Para toda configuración no terminal, solo existe una única configuración siguiente
- Para toda configuración terminal, no existe configuración siguiente
- Transitar actualiza el estado y la cadena en cada paso. Transitar consume un símbolo de la cadena en cada paso (siempre uno)
- $L(AFD) \in N \to |K| > |L(AFD)| \wedge |K| > max\{|w| \mid w \in L(AFD)\}$, ya que no puede contener bucles infinitos

AFND

- Consume uno o varios símbolos en cada transición
- Para toda configuración terminal o no terminal, puede existir cero, una o más configuraciones siguientes
- Transitar **NO** es función
- Transiciones Épsilon donde no consumen ningún carácter de la cadena

AFDM

Denominamos AFDM a aquél autómata AFD con el menor número de estados capaz de representar un determinado lenguaje L. Todos los demás autómatas finitos tienen igual o mayor número de estados que este.

- $M \in AFDM \wedge L(M) = L \to |K_M| = |\pi_L|$ El número de clases de equivalencia de un lenguaje L es igual al número de estados de un AFDM que lo reconoce

APND

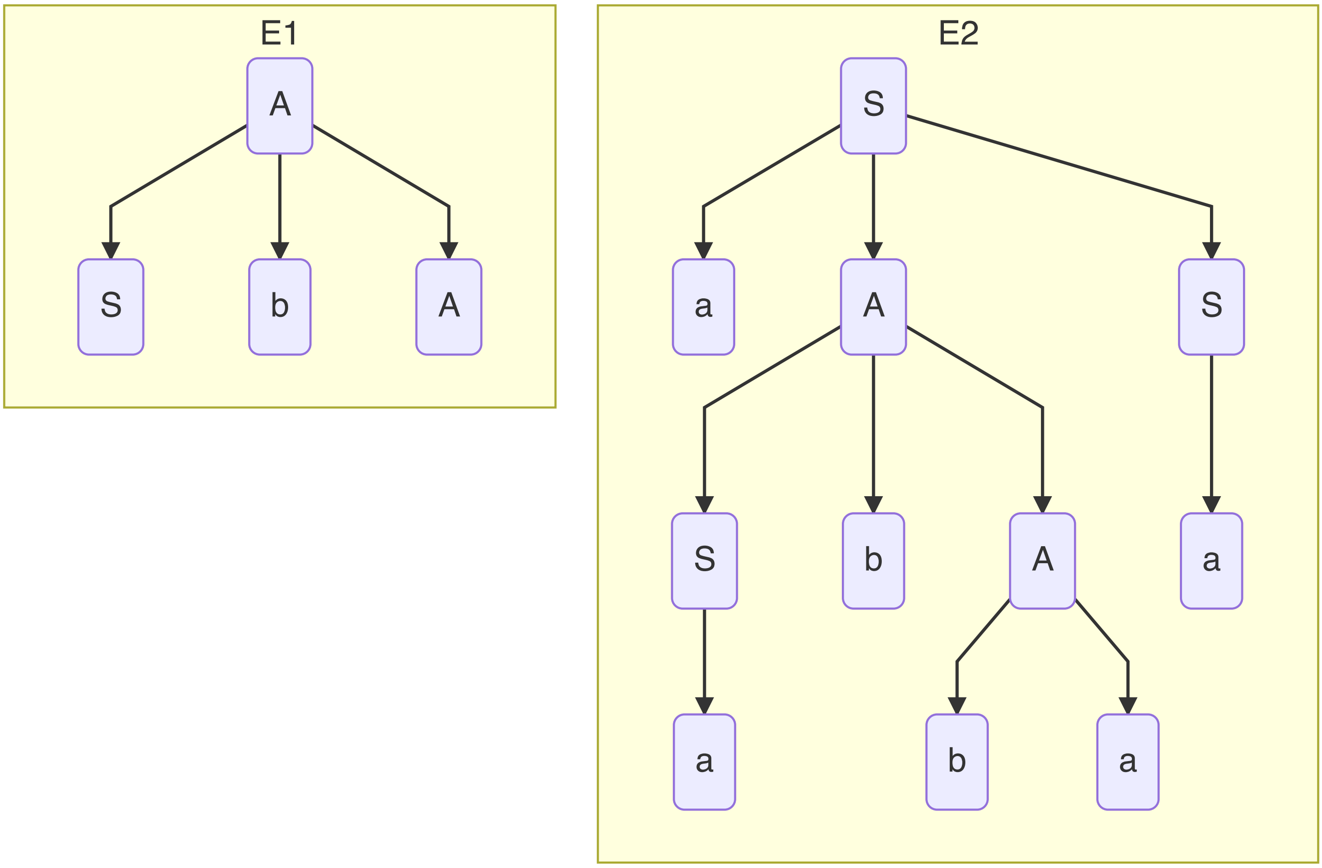
- Transita en función de la cadena, la pila y el estado
- Si transita sin utilizar la pila, entonces el lenguaje es regular

Lenguajes de contexto libre

Definiciones iniciales

- **Sub árbol A de derivación**: Cada nodo tiene un símbolo de V, donde la etiqueta raíz es A
- **Producto de sub árbol A de derivación**: Recorrido en preorden de las etiquetas de las hojas del sub árbol a. Todo producto de un árbol es una forma sentencial perteneciente a V^+

- **A-derivación**: Secuencia finita de producciones directas obtenidas a partir de A aplicando las reglas del sub árbol A
- **Derivación extrema izquierda/derecha**: Si en cada paso de una derivación se aplica una regla al no terminal más a la izquierda/derecha entonces la derivación es extrema izquierda/derecha



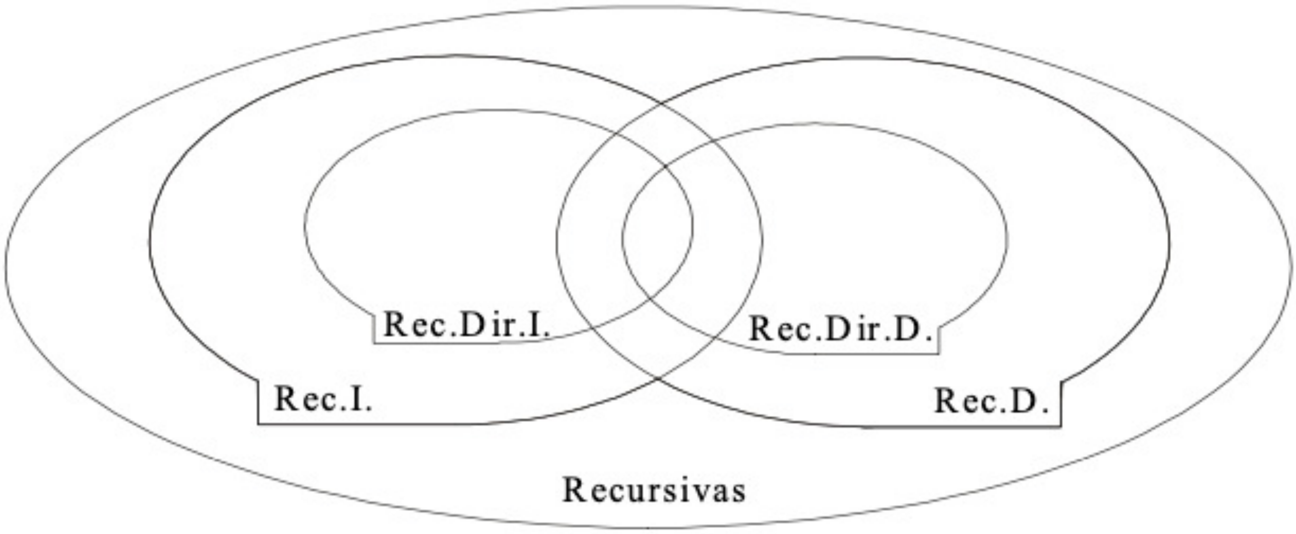
	E1	E2
Producto	SbA	$aabbaa$
Derivaciones	$A \Rightarrow SbA$	$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aSbbaS \Rightarrow aSbbaa \Rightarrow aabbaa$
		$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$
		$S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$

- **Gramática ambigua**: Diremos que G es ambigua si es producto de más de un árbol de derivación (más de un árbol de G genera la cadena). Esta propiedad es **indecidible**
- **Lenguaje inherentemente ambiguo**: Un lenguaje de contexto libre L es ambiguo si todas las gramáticas que lo generan son ambiguas

Recursividad

Nombre	Abv	Fórmula	Restricciones
Recursiva directa izquierda	RDI	$\exists A \in N \Rightarrow A\alpha$	$\alpha \in V^*$
Recursiva directa derecha	RDD	$\exists A \in N \Rightarrow \alpha A$	$\alpha \in V^*$
Recursiva izquierda	RI	$\exists A \in N \Rightarrow^+ A\alpha$	$\alpha \in V^*$
Recursiva derecha	RD	$\exists A \in N \Rightarrow^+ \alpha A$	$\alpha \in V^*$
Recursiva	R	$\exists A \in N \Rightarrow^+ \alpha A\beta$	$\alpha, \beta \in V^*$

> Si G es CL, $\exists G'$ no RDI/RI tal que $G \equiv G'$. **SOLO CON LA IZQUIERDA**
>
> Sii G es RDD y RDI entonces G no es regular

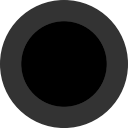


Simplificación de GLC

- **Símbolo útil**: Un símbolo $X \in V$ es útil sii $\exists \alpha, \beta \in V^*, w \in T^+ | S \Rightarrow^* \alpha X \beta \Rightarrow^* w$
- **Símbolo terminable**: Un símbolo $X \in V$ es terminable sii a partir de él podemos encontrar una cadena w terminal (A partir de ese solo)
- **Símbolo accesible**: Un símbolo $X \in V$ es accesible sii a partir del axioma podemos encontrar ese símbolo en alguna derivación (acompañado por otros símbolos o solo)
- **Gramática propia**: Una gramática $G \in GCL$ es propia sii no es recursiva izquierda y no tiene símbolos inútiles

> Que un símbolo sea terminable y accesible no significa que sea útil
>
> $|S_{util}| \neq |T|$
>
> Si $G \in GLC \rightarrow \exists G' \in GLC_{propia} \wedge G \equiv G'$
>
> Si $G \in GLC \rightarrow \exists G'\ no\ tiene\ reglas\ unitarias \wedge G \equiv G'$
>
> No todas las gramáticas GLC tienen otra gramática equivalente propia

Formas normales



Toda GLC se puede expresar como otra en forma normal y viceversa

- De Chomsky: Toda regla de producción genera o bien 2 símbolos no terminales o bien 1 símbolo terminal
- De Greibach: Toda regla de producción genera un símbolo terminal y una cadena no terminal ($\in N^*$)

> $G \in FNC \rightarrow G \notin RI$. Si es una FNC no es recursiva por la izquierda

Propiedades

- L.2 es cerrado para las operaciones de unión, concatenación y estrella de Kleene
- L.2 no es cerrado para las operaciones de intersección y complemento