Tema 5: Autómatas con pila

Autómatas y Lenguajes Formales

Dpto. de Ingeniería de la Información y las Comunicaciones

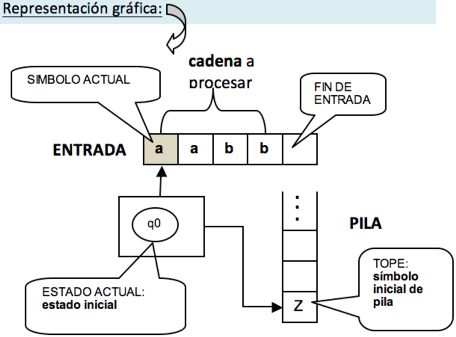
UNIVERSIDAD DE MURCIA

Modelo de autómata con pila

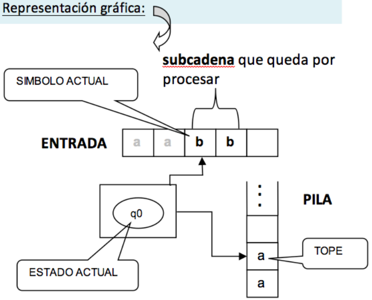
Un **autómata con pila** (ap) es en esencia un autómata finito aumentado con una **memoria tipo pila**, con lo cual, es un tipo de máquina teórica **más potente que un AF**.

Ej.- las cadenas del tipo $a^n b^n$ no forman un lenguaje regular y sin embargo pueden ser validadas por un ap.

Configuración inicial de un AP con cadena de entrada **aabb.**
Representación matemática: **(q0,aabb,Z)**
Representación gráfica:



Configuración intermedia tras extraer Z, haber leído las **a's** y haberlas insertado en la pila.
Representación matemática: **(q0,bb,aa)**
Representación gráfica:

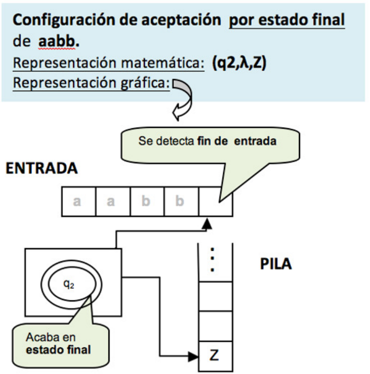


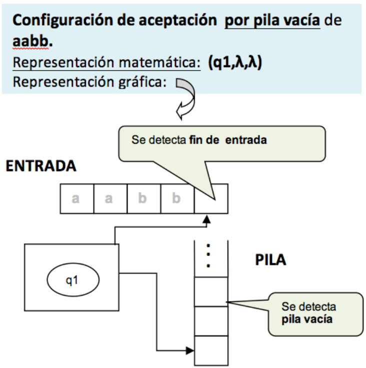
Funcionamiento de un AP

- Los cambios de estado no sólo dependen del **estado actual** y del **símbolo actual**, sino también del **tope de la pila.**
- Teniendo en cuenta esos 3 tres parámetros se aplica la regla de transición correspondiente: **cambia de estado, avanza o no en la entrada y modifica el contenido de la pila.**
- Los ap pueden aceptar cadenas por estado final (ap tipo ef) o por pila vacía (ap tipo pv).

Funcionamiento de un AP

- Los cambios de estado no sólo dependen del **estado actual** y del **símbolo actual**, sino también del **tope de la pila.**
- Teniendo en cuenta esos 3 tres parámetros se aplica la regla de transición correspondiente: **cambia de estado, avanza o no en la entrada y modifica el contenido de la pila.**
- Los ap pueden aceptar cadenas por estado final (ap tipo ef) o por pila vacía (ap tipo pv).





Definición formal de un AP

Un **autómata con pila** (*pushdown automaton, PDA*) es un modelo de máquina teórica que se define matemáticamente mediante una 7-upla $M = (Q, V, \Sigma, \delta, q_0, Z, F)$ donde:

- Q es un conjunto finito de estados
- V es el alfabeto de entrada
- Σ es el **alfabeto de pila**
- q_0 es el estado inicial
- $Z \in \Sigma$ es el **símbolo inicial de la pila**
- $F \subseteq Q$ es el conjunto de estados finales (puede ser vacío)
- $\delta : Q \times (V \cup \{\lambda\}) \times \Sigma \longrightarrow \mathcal{P}(Q \times \Sigma^*)$ es la **función de transición.**

Un autómata con pila es, por defecto, **no determinista**.

Ej.- por **reglas de transición de múltiples opciones** como:

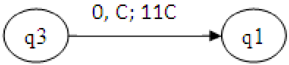
$$\delta(q_3, 0, C) = \{(q_1, 11C), (q_2, \lambda)\}$$

o por λ-**transiciones** como: $\delta(q_3, \lambda, C) = \{(q_1, 11C)\}$

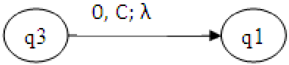
Tipos de reglas de transición. Representación gráfica

Regla con lectura de símbolo, con o sin inserción en pila y una opción para cambio de estado: $\delta(q, a, A) = \{(q', \alpha)\}$
abreviadamente, $\delta(q, a, A) = (q', \alpha)$

Regla **δ (q3, 0, C)= (q1, 11C)**



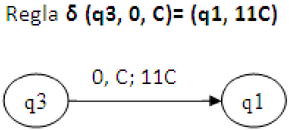
Regla **δ (q3, 0, C)= (q1, λ)**

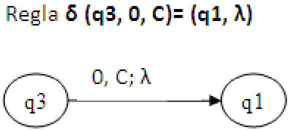


Ej.-

Tipos de reglas de transición. Representación gráfica

Regla con lectura de símbolo, con o sin inserción en pila y una opción para cambio de estado: $\boxed{\delta(q, a, A) = \{(q', \alpha)\}}$
abreviadamente, $\delta(q, a, A) = (q', \alpha)$





Ej.-

Si la **configuración actual** es $(q_3, 0101, C1C)$ entonces:

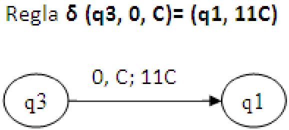
1. Aplicando la regla $\delta(q_3, 0, C) = (q_1, 11C)$ se tiene que:

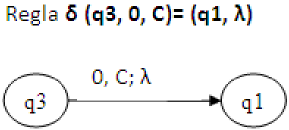
$$(q_3, 0101, C1C) \Rightarrow (q_1, 101, 11C1C)$$

Tipos de reglas de transición. Representación gráfica

Regla con lectura de símbolo, con o sin inserción en pila y una opción para cambio de estado: $\delta(q, a, A) = \{(q', \alpha)\}$

abreviadamente, $\delta(q, a, A) = (q', \alpha)$





Ej.-

Si la **configuración actual** es $(\mathrm{q}_3, 0101, \mathrm{C1C})$ entonces:

1. Aplicando la regla $\delta(q_3, 0, C) = (q_1, 11C)$ se tiene que:

$$(q_3, 0101, C1C) \Rightarrow (q_1, 101, 11C1C)$$

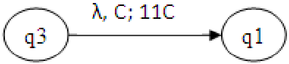
2. Aplicando la regla $\delta(q_3, 0, C) = (q_1, \lambda)$ se tiene que:

$$(q_3, 0101, C1C) \Rightarrow (q_1, 101, 1C)$$

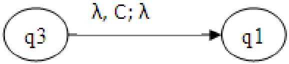
Tipos de reglas de transición. Representación gráfica

Regla sin lectura de símbolo (λ-transición), con o sin inserción en pila y una opción para cambio de estado: $\delta(q, \lambda, A) = (q', \alpha)$

Regla **δ (q3, λ, C)= (q1, 11C)**



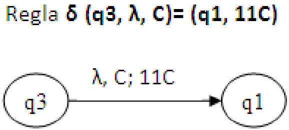
Regla **δ (q3, λ, C)= (q1, λ)**

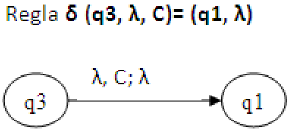


Ej.-

Tipos de reglas de transición. Representación gráfica

Regla sin lectura de símbolo ($\underline{\lambda\text{-transición}}$), con o sin inserción en pila y una opción para cambio de estado: $\delta(q, \lambda, A) = (q', \alpha)$





Ej.-

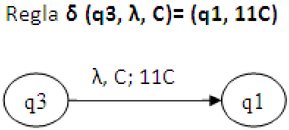
Si la **configuración actual** es $(q_3, 0101, C1C)$ entonces:

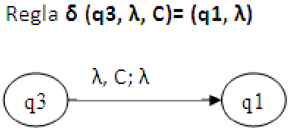
1. Aplicando la regla $\delta(q_3, \lambda, C) = (q_1, 11C)$ se tiene que:

$$(q_3, \underline{0101}, C1C) \Rightarrow (q_1, \underline{0101}, 11C1C)$$

Tipos de reglas de transición. Representación gráfica

Regla sin lectura de símbolo (λ-transición), con o sin inserción en pila y una opción para cambio de estado: $\delta(q, \lambda, A) = (q', \alpha)$





Ej.-

Si la **configuración actual** es $(q_3, 0101, C1C)$ entonces:

1. Aplicando la regla $\delta(q_3, \lambda, C) = (q_1, 11C)$ se tiene que:

$$(q_3, \underline{0101}, C1C) \Rightarrow (q_1, \underline{0101}, 11C1C)$$

2. Aplicando la regla $\delta(q_3, \lambda, C) = (q_1, \lambda)$ se tiene que:

$$(q_3, \underline{0101}, C1C) \Rightarrow (q_1, \underline{0101}, 1C)$$

Tipos de reglas de transición. Representación gráfica

Regla sin lectura de símbolo (λ-transición), con o sin inserción en pila y una opción para cambio de estado: $\delta(q, \lambda, A) = (q', \alpha)$

No determinismo: las λ-transiciones pueden ser causa de no determinismo en los cálculos.

Ej.-

Si la **configuración actual** es $(q_3, 0101, C1C)$ y tenemos definidas las reglas $\delta(q_3, \lambda, C) = (q_1, 11C)$ y $\delta(q_3, 0, C) = (q_4, CC)$, entonces se pueden alcanzar dos configuraciones distintas en un paso de cálculo:

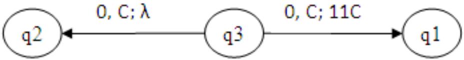
$$(q_3, 0101, C1C) \begin{cases} \Rightarrow (q_1, 0101, 11C1C) \ \ [\text{por } \delta(q_3, \lambda, C) = (q_1, 11C)] \\ \\ \Rightarrow (q_4, 101, CC1C) \ \ [\text{por } \delta(q_3, 0, C) = (q_4, CC)] \end{cases}$$

Tipos de reglas de transición. Representación gráfica

Regla con lectura de símbolo, con o sin inserción en pila y $k > 0$ opciones para cambio de estado:

$$\delta(q, a, A) = \{(q_{i1}, \alpha_1), (q_{i2}, \alpha_2), \ldots, (q_{ik}, \alpha_k)\}$$

Ej.- $\delta(q_3, 0, C) = \{(q_2, \lambda), (q_1, 11C)\}$



No determinismo: estas reglas son causa de no determinismo

Si la **configuración actual** es $(q_3, 0101, C1C)$ entonces:

$$(q_3, 0101, C1C) \begin{cases} \Rightarrow (q_2, 101, 1C) \text{ [por opción } (q_2, \lambda) \in \delta(q_3, 0, C) \text{]} \\ \\ \Rightarrow (q_1, 101, 11C1C) \text{ [por opción } (q_1, 11C) \in \delta(q_3, 0, C) \text{]} \end{cases}$$

Tipos de reglas de transición. Representación gráfica

Regla sin lectura de símbolo, con o sin inserción en pila y $k > 0$ opciones para cambio de estado:

$$\delta(q, \lambda, A) = \{(q_{i1}, \alpha_1), (q_{i2}, \alpha_2), \ldots, (q_{ik}, \alpha_k)\}$$

Ej.- $\delta(q_3, \lambda, C) = \{(q_2, \lambda), (q_1, 11C)\}$

No determinismo: estas reglas son causa de no determinismo.
Si la **configuración actual** es $(q_3, 0101, C1C)$ entonces:

$$(q_3, 0101, C1C) \begin{cases} \Rightarrow (q_2, 0101, 1C) \text{ [por opción } (q_2, \lambda) \in \delta(q_3, \lambda, C)] \\ \Rightarrow (q_1, 0101, 11C1C) \text{ [por opción } (q_1, 11C) \in \delta(q_3, \lambda, C)] \end{cases}$$

Lenguaje aceptado por un AP

Informalmente, **el lenguaje aceptado por un AP es el conjunto de cadenas que el AP puede aceptar**.
Formalmente, definimos el lenguaje aceptado por un ap dependiendo del tipo de ap:

- ap tipo pv: no tienen estados finales ($F = \emptyset$) y **aceptan cadenas por pila vacía** (se detecta fin de entrada y pila vacía).

$$\boxed{L_{\text{pv}}(M) = \{w \in V^* \mid (q_0, w, Z) \Rightarrow^* (q, \lambda, \lambda)\}}$$

- ap tipo ef: tienen estados finales y **aceptan cadenas por estado final** (se detecta fin de entrada y para en estado final).

$$\boxed{L_{\text{ef}}(M) = \{w \in V^* \mid (q_0, w, Z) \Rightarrow^* (q_F, \lambda, \gamma),\ q_F \in F, \gamma \in \Sigma^*\}}$$

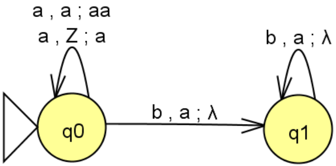
Nota: Si el ap acepta por estado final entonces es posible aceptar λ sin aplicar ninguna regla; esto pasa cuando q_0 es final.
Si acepta por pila vacía entonces tendrá que aplicar al menos una regla.

Ejemplo: deducción del lenguaje aceptado por AP tipo PV

Función de transición

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

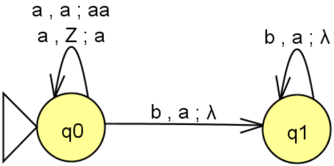
Diagrama de transición



Función de transición de M_1

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición

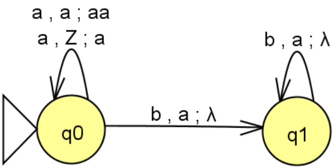


1. La única regla aplicable desde una configuración inicial es la regla 1 $\Rightarrow$ las cadenas deben comenzar por a.

Función de transición de M_1

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición

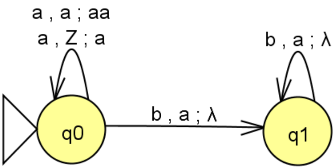


1. La única regla aplicable desde una configuración inicial es la regla 1 $\Rightarrow$ las cadenas deben comenzar por a.
2. **La regla 2 permite que por cada a que se lea de la cadena de entrada se apile una a.**

Función de transición de M_1

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición

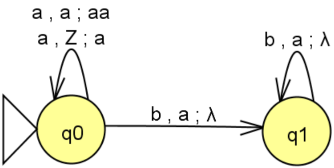


1. La única regla aplicable desde una configuración inicial es la regla 1 $\Rightarrow$ las cadenas deben comenzar por *a*.
2. La regla 2 permite que por cada *a* que se lea de la cadena de entrada se apile una *a*.
3. **Por la regla 3 cambia de estado cuando lee una *b* y hay una *a* en la cima de la pila $\Rightarrow$ sólo b's tras las a's.**

Función de transición de M_1

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición

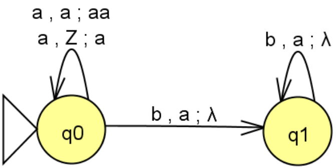


1. La única regla aplicable desde una configuración inicial es la regla 1 $\rightrightarrows$ las cadenas deben comenzar por a.
2. La regla 2 permite que por cada a que se lea de la cadena de entrada se apile una a.
3. Por la regla 3 cambia de estado cuando lee una b y hay una a en la cima de la pila $\rightrightarrows$ sólo b's tras las a's.
4. **La regla 4 permite que por cada b que se lea de la cadena de entrada se extraiga una a de la pila.**

Función de transición de M_1

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición



1. La única regla aplicable desde una configuración inicial es la regla 1 $\Rightarrow$ las cadenas deben comenzar por a.
2. La regla 2 permite que por cada a que se lea de la cadena de entrada se apile una a.
3. Por la regla 3 cambia de estado cuando lee una b y hay una a en la cima de la pila $\Rightarrow$ sólo b's tras las a's.
4. La regla 4 permite que por cada b que se lea de la cadena de entrada se extraiga una a de la pila.

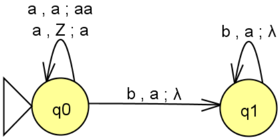
   La pila se vacía sólo cuando se han leído el mismo número de $a's$ que de $b's$ $\Rightarrow$ $\boxed{L_{\mathbf{pv}}(M_1) = \{a^n b^n \mid n > 0\}}$

Ejemplo: cálculos que aceptan cadenas

Función de transición de M_1

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición



Ej.- La cadena de entrada *aabb* es aceptada por el cálculo:

$$(q_0, aabb, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, abb, a)$$

Ejemplo: cálculos que aceptan cadenas

Función de transición de M_1

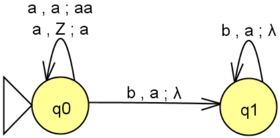
$1: \ \delta(q_0, a, Z) = (q_0, a)$

$2: \ \delta(q_0, a, a) = (q_0, aa)$

$3: \ \delta(q_0, b, a) = (q_1, \lambda)$

$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición



Ej.- La cadena de entrada *aabb* es aceptada por el cálculo:

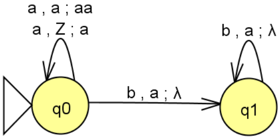
$$(q_0, aabb, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, abb, a) \overbrace{\Rightarrow}^{Tr2} (q_0, bb, aa)$$

Ejemplo: cálculos que aceptan cadenas

Función de transición de M_1

$1: \ \delta(q_0, a, Z) = (q_0, a)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición



Ej.- La cadena de entrada *aabb* es aceptada por el cálculo:

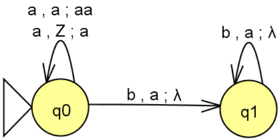
$$(q_0, aabb, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, abb, a) \overbrace{\Rightarrow}^{Tr2} (q_0, bb, aa) \overbrace{\Rightarrow}^{Tr3} (q_1, b, a)$$

Ejemplo: cálculos que aceptan cadenas

Función de transición de M_1

$$
\begin{aligned}
1 &: \ \delta(q_0, a, Z) = (q_0, a) \\
2 &: \ \delta(q_0, a, a) = (q_0, aa) \\
3 &: \ \delta(q_0, b, a) = (q_1, \lambda) \\
4 &: \ \delta(q_1, b, a) = (q_1, \lambda)
\end{aligned}
$$

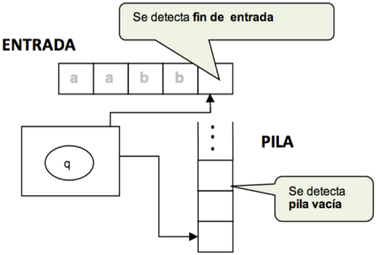
Diagrama de transición



Ej.- La cadena de entrada *aabb* es aceptada por el cálculo:

$$(q_0, aabb, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, abb, a) \overbrace{\Rightarrow}^{Tr2} (q_0, bb, aa) \overbrace{\Rightarrow}^{Tr3}$$

$$(q_1, b, a) \overbrace{\Rightarrow}^{Tr4} (q_1, \lambda, \lambda)$$



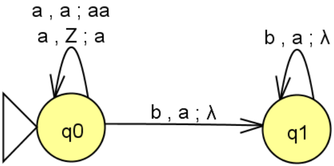
Ejemplo: cálculos que rechazan cadenas

$L_{\mathrm{pv}}(M_1) = \{a^n b^n \mid n > 0\}$

Función de transición de M_1

1 : $\delta(q_0, a, Z) = (q_0, a)$
2 : $\delta(q_0, a, a) = (q_0, aa)$
3 : $\delta(q_0, b, a) = (q_1, \lambda)$
4 : $\delta(q_1, b, a) = (q_1, \lambda)$

Diagrama de transición



Ej.- *abb* no es aceptada: $(q_0, abb, Z) \Rightarrow (q_0, bb, a) \Rightarrow (q_1, b, \lambda)$

Ej.- *aab* no es aceptada:

$$(q_0, aab, Z) \Rightarrow (q_0, ab, a) \Rightarrow (q_0, b, aa) \Rightarrow (q_1, \lambda, a)$$

Ej.- λ no es aceptada porque a partir de la configuración inicial (q_0, λ, Z) no se puede aplicar ninguna regla (no vacía la pila).

Aceptar por pila vacía es equivalente a aceptar por estado final

Teorema Si un lenguaje es aceptado por estado final por un autómata con pila M_f entonces también puede ser aceptado por pila vacía por otro autómata M_v y al contrario.

Ejemplo: de AP que acepta por pila vacía a AP por estado final

M_1 acepta $\{a^n b^n \mid n > 0\}$ por pila vacía y M_2 lo acepta por estado final.

Función de transición de M_1		**Función de transición** de M_2
$F_1 = \varnothing$		$F_2 = \{q_2\}$
$1: \ \delta(q_0, a, Z) = (q_0, a)$	$\Rightarrow$	$1: \ \delta(q_0, a, Z) = (q_0, aZ)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$		$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$		$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$		$4: \ \delta(q_1, b, a) = (q_1, \lambda)$
		$5: \ \delta(q_1, \lambda, Z) = (q_2, Z)$

Diseño con autómatas con pila. AP determinista

Corrección del diseño

Para resolver un problema de procesamiento de cadenas usando como modelo un AP hay que asegurarse de que **el AP es correcto**:

1. El ap no es demasiado estricto: acepta todas las cadenas consideradas válidas.
2. El M no es demasiado general: rechaza cualquier cadena incorrecta.

Diseño con autómatas con pila. AP determinista

Corrección del diseño

Para resolver un problema de procesamiento de cadenas usando como modelo un AP hay que asegurarse de que **el AP es correcto**:

1. El ap no es demasiado estricto: acepta todas las cadenas consideradas válidas.
2. El M no es demasiado general: rechaza cualquier cadena incorrecta.

Intentar que el AP sea determinista

Un **algoritmo de simulación** de un AP determinista tiene un tiempo de ejecución de orden lineal: $O(n)$.

Un ap es **determinista** si no es posible que a partir de una configuración se puedan alcanzar dos o más configuraciones distintas en un paso de cálculo. En otro caso decimos que el ap es **no determinista**.

AP determinista

Intentar que el AP sea determinista

Un **algoritmo de simulación** de un AP determinista tiene un tiempo de ejecución de orden lineal: $O(n)$.

Un ap es **determinista** si no es posible que a partir de una configuración se puedan alcanzar dos o más configuraciones distintas en un paso de cálculo. En otro caso decimos que el ap es **no determinista**.

Condiciones para que un AP sea determinista

1. **No tiene reglas de transición de múltiples opciones**, tipo $\delta(q, a, A) = \{(q_{i1}, \alpha_1), \ldots, (q_{ik}, \alpha_k)\}$, o bien, $\delta(q, \lambda, A) = \{(q_{i1}, \alpha_1), \ldots, (q_{ik}, \alpha_k)\}$.
2. **No hay definida simultáneamente una transición con un símbolo $a \in V$ y con λ, para un mismo estado q y tope de pila** A. Es decir, si $\delta(q, a, A) \neq \varnothing$ entonces debe ser $\delta(q, \lambda, A) = \varnothing$ y al contrario, si $\delta(q, \lambda, A) \neq \varnothing$ entonces debe ser $\delta(q, a, A) = \varnothing$, para cualquier $a \in V$.

Ejemplo: obtener un AP que acepte $\{a^n b^n \mid n \geq 0\}$ *(contiene a* λ*)*

M_{v0} acepta $\{a^n b^n \geq 0\}$ por pila vacía y M_{f0} por estado final.

Ejemplo: obtener un AP que acepte $\{a^n b^n \mid n \geq 0\}$ (contiene a λ)

M_{v0} acepta $\{a^n b^n \geq 0\}$ por pila vacía y M_{f0} por estado final.

$F_{v0} = \varnothing$ **Función de transición** de M_{v0} $0: \ \delta(q_0, \lambda, Z) = (q_1, \lambda)$ $1: \ \delta(q_0, a, Z) = (q_0, a)$ $2: \ \delta(q_0, a, a) = (q_0, aa)$ $3: \ \delta(q_0, b, a) = (q_1, \lambda)$ $4: \ \delta(q_1, b, a) = (q_1, \lambda)$	$F_{f0} = \{q_2\}$ **Función de transición** de M_{f0} $0: \ \delta(q_0, \lambda, Z) = (q_2, Z)$ $1: \ \delta(q_0, a, Z) = (q_0, aZ)$ $2: \ \delta(q_0, a, a) = (q_0, aa)$ $3: \ \delta(q_0, b, a) = (q_1, \lambda)$ $4: \ \delta(q_1, b, a) = (q_1, \lambda)$ $5: \ \delta(q_1, \lambda, Z) = (q_2, Z)$

Ejemplo: obtener un AP que acepte $\{a^n b^n \mid n \geq 0\}$

M_{v0} acepta $\{a^n b^n \geq 0\}$ por pila vacía y M_{f0} por estado final.

Función de transición de M_{v0}	**Función de transición** de M_{f0}
$F_{v0} = \varnothing$	$F_{f0} = \{q_2\}$
$0: \ \delta(q_0, \lambda, Z) = (q_1, \lambda)$	$0: \ \delta(q_0, \lambda, Z) = (q_2, Z)$
$1: \ \delta(q_0, a, Z) = (q_0, a)$	$1: \ \delta(q_0, a, Z) = (q_0, aZ)$
$2: \ \delta(q_0, a, a) = (q_0, aa)$	$2: \ \delta(q_0, a, a) = (q_0, aa)$
$3: \ \delta(q_0, b, a) = (q_1, \lambda)$	$3: \ \delta(q_0, b, a) = (q_1, \lambda)$
$4: \ \delta(q_1, b, a) = (q_1, \lambda)$	$4: \ \delta(q_1, b, a) = (q_1, \lambda)$
	$5: \ \delta(q_1, \lambda, Z) = (q_2, Z)$

Son **no deterministas**:

Múltiples configuraciones en M_{v0}

$$(q_0, aabb, Z) \begin{cases} \Rightarrow (q_1, aabb, \lambda) \\ \Rightarrow (q_0, abb, a) \end{cases}$$

Múltiples configuraciones en M_{f0}

$$(q_0, aabb, Z) \begin{cases} \Rightarrow (q_2, aabb, Z) \\ \Rightarrow (q_0, abb, aZ) \end{cases}$$

Ejemplo: AP determinista para $\{a^n b^n \mid n \geq 0\}$

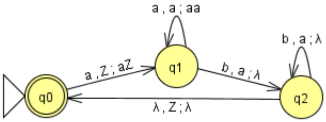
<u>**Solución:**</u> para evitar el no determinismo al aceptar λ podemos hacer que **el estado inicial sea un estado final**.

Función de transición de M_{detf0}

$F = \{q_0\}$

1. $\delta(q_0, a, Z) = (q_1, aZ)$
2. $\delta(q_1, a, a) = (q_1, aa)$
3. $\delta(q_1, b, a) = (q_2, \lambda)$
4. $\delta(q_2, b, a) = (q_2, \lambda)$
5. $\delta(q_2, \lambda, Z) = (q_0, \lambda)$

Diagrama de transición:



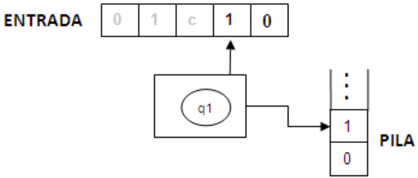
- La cadena $\lambda = a^0 b^0$ es aceptada por el cálculo en cero pasos: $(q_0, \lambda, Z) \Rightarrow^* (q_0, \lambda, Z)$.
- Las cadenas del tipo $a^n b^n$, con $n > 0$, se aceptan: $(q_0, a^n b^n, Z) \Rightarrow (q_1, a^{n-1} b^n, aZ) \Rightarrow^* (q_1, b^n, a^n Z) \Rightarrow^* (q_2, \lambda, Z) \Rightarrow (q_0, \lambda, \lambda)$

Ejemplo: AP determinista para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

Idea para diseñar el autómata:

- Mientras no se lea una c en la entrada, vamos introduciendo los símbolos leídos en la pila.
- Cuando se lee una c cambiamos de estado, para comenzar a comparar la cadena de la pila con lo que queda de la entrada.
- La cadena de entrada será aceptada si y sólo si la subcadena que queda por leer tras la c coincide exactamente con la cadena de la pila $\rightrightarrows$ avanzar y extraer si coincide símbolo y tope.

Ej.- si la cadena de entrada es 01c10, entonces la configuración del ap tras haber leído la c sería:



Ejemplo: AP <u>determinista</u> para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por <u>pila vacía:</u>

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Ejemplo: AP <u>determinista</u> para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por <u>pila vacía:</u>

1. $\delta(q_0, 0, Z) = (q_0, 0)$	7. $\delta(q_0, 1, 1) = (q_0, 11)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$	8. $\delta(q_0, c, 0) = (q_1, 0)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$	9. $\delta(q_0, c, 1) = (q_1, 1)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$	10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$	11. $\delta(q_1, 1, 1) = (q_1, \lambda)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$	

Test de prueba con cadenas <u>válidas:</u>

$$(q_0, c, Z) \overbrace{\Rightarrow}^{Tr3} (q_1, \lambda, \lambda) \quad [\text{ acepta } c \text{ }]$$

Ejemplo: AP determinista para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por pila vacía:

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Test de prueba con cadenas válidas:

$$(q_0, c, Z) \overbrace{\Rightarrow}^{Tr3} (q_1, \lambda, \lambda) \quad [\text{ acepta } c \text{ }]$$

$$(q_0, 01c10, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, 1c10, 0)$$

Ejemplo: AP ***determinista*** *para* $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por pila vacía:

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Test de prueba con cadenas válidas:

$$(q_0, c, Z) \overbrace{\Rightarrow}^{Tr3} (q_1, \lambda, \lambda) \quad [\text{ acepta } c\ ]$$

$$(q_0, 01c10, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, 1c10, 0) \overbrace{\Rightarrow}^{Tr5} (q_0, c10, 10)$$

Ejemplo: AP determinista para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por pila vacía:

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Test de prueba con cadenas válidas:

$(q_0, c, Z) \overbrace{\Rightarrow}^{Tr3} (q_1, \lambda, \lambda)$ [acepta c]

$(q_0, 01c10, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, 1c10, 0) \overbrace{\Rightarrow}^{Tr5} (q_0, c10, 10) \overbrace{\Rightarrow}^{Tr9} (q_1, 10, 10)$

Ejemplo: AP determinista para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por pila vacía:

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Test de prueba con cadenas válidas:

$(q_0, c, Z) \overbrace{\Rightarrow}^{Tr3} (q_1, \lambda, \lambda)$ [acepta c]

$(q_0, 01c10, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, 1c10, 0) \overbrace{\Rightarrow}^{Tr5} (q_0, c10, 10) \overbrace{\Rightarrow}^{Tr9}$

$(q_1, 10, 10) \overbrace{\Rightarrow}^{Tr11} (q_1, 0, 0)$

Ejemplo: AP determinista para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por pila vacía:

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Test de prueba con cadenas válidas:

$(q_0, c, Z) \overbrace{\Rightarrow}^{Tr3} (q_1, \lambda, \lambda)$ [acepta c]

$(q_0, 01c10, Z) \overbrace{\Rightarrow}^{Tr1} (q_0, 1c10, 0) \overbrace{\Rightarrow}^{Tr5} (q_0, c10, 10) \overbrace{\Rightarrow}^{Tr9}$

$(q_1, 10, 10) \overbrace{\Rightarrow}^{Tr11} (q_1, 0, 0) \overbrace{\Rightarrow}^{Tr10} (q_1, \lambda, \lambda)$ [acepta $01c10$]

Ejemplo: AP determinista para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por pila vacía:

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Test de prueba con cadenas incorrectas:

- λ no es aceptada porque $\delta(q_0, \lambda, Z)$ no está definida.

Ejemplo: AP determinista para $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$

M_{wcwr} que acepta L_{wcwr} por pila vacía:

1. $\delta(q_0, 0, Z) = (q_0, 0)$
2. $\delta(q_0, 1, Z) = (q_0, 1)$
3. $\delta(q_0, c, Z) = (q_1, \lambda)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = (q_0, 11)$
8. $\delta(q_0, c, 0) = (q_1, 0)$
9. $\delta(q_0, c, 1) = (q_1, 1)$
10. $\delta(q_1, 0, 0) = (q_1, \lambda)$
11. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Test de prueba con cadenas incorrectas:

- λ no es aceptada porque $\delta(q_0, \lambda, Z)$ no está definida.
- 01c01 no es aceptada:
  $(q_0, 01c01, Z) \Rightarrow (q_0, 1c10, 0) \Rightarrow (q_0, c01, 10) \Rightarrow (q_1, 01, 10)$
- 10c011 no es aceptada:
  $(q_0, 10c011, Z) \Rightarrow (q_0, 0c011, 1) \Rightarrow (q_0, c011, 01) \Rightarrow (q_1, 011, 01)$
  $\Rightarrow (q_1, 11, 1) \Rightarrow (q_1, 1, \lambda)$
- Con la cadena 01c1 termina de leer pero no vacía la pila.

Lenguajes libres del contexto no deterministas

Un lenguaje es **libre del contexto determinista** si existe un ap determinista que lo acepta.
Si no existe un ap determinista, pero sí un ap no determinista que lo acepta, entonces el lenguaje es **libre del contexto no determinista**.

- **No todos los lenguajes libres del contexto pueden ser aceptados por autómatas con pila deterministas.**
  El lenguaje de palíndromos binarios de longitud par es un LLC no determinista:
  $$L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$$
- **Los AP deterministas y no deterministas no son equivalentes**.

Ejemplo: LLC no determinista $L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$

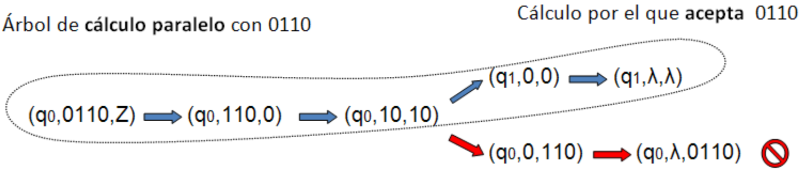
AP no determinista M_{wwr} que acepta L_{wwr} por pila vacía.

1. $\delta(q_0, \lambda, Z) = (q_1, \lambda)$
2. $\delta(q_0, 0, Z) = (q_0, 0)$
3. $\delta(q_0, 1, Z) = (q_0, 1)$
4. $\delta(q_0, 0, 0) = \{(q_0, 00), (q_1, \lambda)\}$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = \{(q_0, 11), (q_1, \lambda)\}$
8. $\delta(q_1, 0, 0) = (q_1, \lambda)$
9. $\delta(q_1, 1, 1) = (q_1, \lambda)$

Ejemplo: LLC no determinista $L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$

AP no determinista M_{wwr} que acepta L_{wwr} por pila vacía.

1. $\delta(q_0, \lambda, Z) = (q_1, \lambda)$
2. $\delta(q_0, 0, Z) = (q_0, 0)$
3. $\delta(q_0, 1, Z) = (q_0, 1)$
4. $\delta(q_0, 0, 0) = \{(q_0, 00), (q_1, \lambda)\}$
5. $\delta(q_0, 1, 0) = (q_0, 10)$
6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 1) = \{(q_0, 11), (q_1, \lambda)\}$
8. $\delta(q_1, 0, 0) = (q_1, \lambda)$
9. $\delta(q_1, 1, 1) = (q_1, \lambda)$



Simulación de AP no determinista: para una cadena de entrada, es necesario explorar las distintas ramas del árbol de cálculo, lo cual supone tener que desarrollar un **algoritmo con backtracking**, de **tiempo exponencial**.

Autómatas con pila y gramáticas libres del contexto

Teorema.- **Un lenguaje puede ser aceptado por un autómata con pila si y sólo si puede ser generado por una gramática libre del contexto** $\rightrightarrows$ los autómatas con pila y las gramáticas libres del contexto son formalismos con la misma expresividad.

Autómatas con pila y gramáticas libres del contexto

Teorema.- **Un lenguaje puede ser aceptado por un autómata con pila si y sólo si puede ser generado por una gramática libre del contexto** $\rightrightarrows$ los autómatas con pila y las gramáticas libres del contexto son formalismos con la misma expresividad.

Método GLCtoAP no determinista

entrada: una glc $G = (V_N, V_T, S, P)$.

salida: un ap M tal que $L_{\mathrm{pv}}(M) = L(G)$.

- $Q := \{q_0, q_1\}$, $F := \emptyset$;
- $V = V_T$; $\Sigma := V_N \cup V_T \cup \{Z\}$;
- La **función de transición** δ se obtiene del siguiente modo:
  1. $\delta(q_0, \lambda, Z) = (q_1, S)$ [introduce símbolo inicial S en pila]
  2. $\delta(q_1, a, a) = (q_1, \lambda)$, $\forall\, a \in V_T$ [extrae cuando coinciden terminales]
  3. Si $A \to \alpha \in P$ entonces $(q_1, \alpha) \in \delta(q_1, \lambda, A)$ [simula aplicación de regla de producción]

Autómatas con pila y gramáticas libres del contexto

Teorema.- **Un lenguaje puede ser aceptado por un autómata con pila si y sólo si puede ser generado por una gramática libre del contexto** $\rightrightarrows$ los autómatas con pila y las gramáticas libres del contexto son formalismos con la misma expresividad.

Método GLCtoAP no determinista

entrada: una glc $G = (V_N, V_T, S, P)$.

salida: un ap M tal que $L_{\mathrm{pv}}(M) = L(G)$.

- $Q := \{q_0, q_1\}$, $F := \emptyset$;
- $V = V_T$; $\Sigma := V_N \cup V_T \cup \{Z\}$;
- La **función de transición** δ se obtiene del siguiente modo:
  1. $\delta(q_0, \lambda, Z) = (q_1, S)$ [introduce símbolo inicial S en pila]
  2. $\delta(q_1, a, a) = (q_1, \lambda)$, $\forall\, a \in V_T$ [extrae cuando coinciden terminales]
  3. Si $A \to \alpha \in P$ entonces $(q_1, \alpha) \in \delta(q_1, \lambda, A)$ [simula aplicación de regla de producción]

Inconveniente: simulación en tiempo exponencial.

Ejemplo: transformar una GLC en un AP

Sea una **GLC** no ambigua que genera expresiones aritméticas:

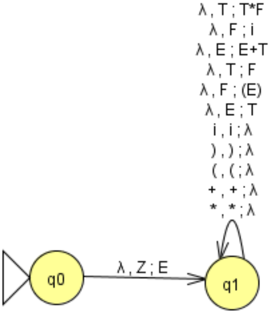
$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid i$$

El AP no determinista que acepta el lenguaje generado por la gramática es:

Función de transición:

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Diagrama de transición:



Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E$$

Cálculo para i + i:

$$(q_0, i + i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i + i, E)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E+T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T*F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$\boxed{E} \overbrace{\Rightarrow}^{E \to E+T} E + T$$

Cálculo para i + i:

$$(q_0, i+i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i+i, \boxed{E}) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i+i, E+T)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E \to E+T} \boxed{E} + T \overbrace{\Rightarrow}^{E \to T} T + T$$

Cálculo para i + i:

$$(q_0, i + i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i + i, E) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i + i, \boxed{E} + T) \overbrace{\Rightarrow}^{Tr1,2} (q_1, i + i, T + T)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E \to E+T} E + T \overbrace{\Rightarrow}^{E \to T} \boxed{T} + T \overbrace{\Rightarrow}^{T \to F} F + T$$

Cálculo para i + i:

$$(q_0, i + i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i + i, E) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i + i, E + T) \overbrace{\Rightarrow}^{Tr1,2}$$
$$(q_1, i + i, \boxed{T} + T) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i + i, F + T)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E \to E+T} E + T \overbrace{\Rightarrow}^{E \to T} T + T \overbrace{\Rightarrow}^{T \to F} \boxed{F} + T \overbrace{\Rightarrow}^{F \to i} i + T$$

Cálculo para i + i:

$$(q_0, i + i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i + i, E) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i + i, E + T) \overbrace{\Rightarrow}^{Tr1,2}$$
$$(q_1, i + i, T + T) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i + i, \boxed{F} + T) \overbrace{\Rightarrow}^{Tr3,2} (q_1, i + i, i + T)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E \to E+T} E + T \overbrace{\Rightarrow}^{E \to T} dT + T \overbrace{\Rightarrow}^{T \to F} \boxed{F} + T \overbrace{\Rightarrow}^{F \to i} i + T$$

Cálculo para i + i:

$$(q_0, i + i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i + i, E) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i + i, E + T) \overbrace{\Rightarrow}^{Tr1,2}$$

$$(q_1, i + i, T + T) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i + i, F + T) \overbrace{\Rightarrow}^{Tr3,2} (q_1, i + i, i + T)$$

$$\overbrace{\Rightarrow}^{Tr6} (q_1, +i, +T)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E\to E+T} E + T \overbrace{\Rightarrow}^{E\to T} dT + T \overbrace{\Rightarrow}^{T\to F} \boxed{F} + T \overbrace{\Rightarrow}^{F\to i} i + T$$

Cálculo para i + i:

$$(q_0, i + i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i + i, E) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i + i, E + T) \overbrace{\Rightarrow}^{Tr1,2}$$
$$(q_1, i + i, T + T) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i + i, F + T) \overbrace{\Rightarrow}^{Tr3,2} (q_1, i + i, i + T)$$
$$\overbrace{\Rightarrow}^{Tr6} (q_1, +i, +T) \overbrace{\Rightarrow}^{Tr7} (q_1, i, T)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E+T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T*F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E \to E+T} E+T \overbrace{\Rightarrow}^{E \to T} T+T \overbrace{\Rightarrow}^{T \to F} F+T \overbrace{\Rightarrow}^{F \to i} i + \boxed{T} \overbrace{\Rightarrow}^{T \to F} i+F$$

Cálculo para i + i:

$$(q_0, i+i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i+i, E) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i+i, E+T) \overbrace{\Rightarrow}^{Tr1,2}$$
$$(q_1, i+i, T+T) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i+i, F+T) \overbrace{\Rightarrow}^{Tr3,2} (q_1, i+i, i+T)$$
$$\overbrace{\Rightarrow}^{Tr6} (q_1, +i, +T) \overbrace{\Rightarrow}^{Tr7} (q_1, i, \boxed{T}) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i, F)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E + T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T * F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E \to E+T} E + T \overbrace{\Rightarrow}^{E \to T} T + T \overbrace{\Rightarrow}^{T \to F} F + T \overbrace{\Rightarrow}^{F \to i} i + T \overbrace{\Rightarrow}^{T \to F} i + \boxed{F} \overbrace{\Rightarrow}^{F \to i} i + i$$

Cálculo para i + i:

$$(q_0, i + i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i + i, E) \overbrace{\Rightarrow}^{Tr1,1} (q_1, i + i, E + T) \overbrace{\Rightarrow}^{Tr1,2}$$
$$(q_1, i + i, T + T) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i + i, F + T) \overbrace{\Rightarrow}^{Tr3,2} (q_1, i + i, i + T)$$
$$\overbrace{\Rightarrow}^{Tr6} (q_1, +i, +T) \overbrace{\Rightarrow}^{Tr7} (q_1, i, T) \overbrace{\Rightarrow}^{Tr2,2} (q_1, i, \boxed{F}) \overbrace{\Rightarrow}^{Tr3,2} (q_1, i, i)$$

Ejemplo: un cálculo simula una derivación más a la izquierda

0. $\delta(q_0, \lambda, Z) = (q_1, E)$
1. $\delta(q_1, \lambda, E) = \{(q_1, E+T), (q_1, T)\}$
2. $\delta(q_1, \lambda, T) = \{(q_1, T*F), (q_1, F)\}$
3. $\delta(q_1, \lambda, F) = \{(q_1, (E)), (q_1, i)\}$
4. $\delta(q_1, (, () = (q_1, \lambda)$
5. $\delta(q_1,),)) = (q_1, \lambda)$
6. $\delta(q_1, i, i) = (q_1, \lambda)$
7. $\delta(q_1, +, +) = (q_1, \lambda)$
8. $\delta(q_1, *, *) = (q_1, \lambda)$

Derivación más a la izquierda para i + i:

$$E \overbrace{\Rightarrow}^{E\to E+T} E+T \overbrace{\Rightarrow}^{E\to T} T+T \overbrace{\Rightarrow}^{T\to F} F+T \overbrace{\Rightarrow}^{F\to i} i+T \overbrace{\Rightarrow}^{T\to F} i+F \overbrace{\Rightarrow}^{F\to i} i+i$$

Cálculo para i + i:

$$(q_0, i+i, Z) \overbrace{\Rightarrow}^{Tr0} (q_1, i+i, E) \overbrace{\Rightarrow}^{Tr1{,}1} (q_1, i+i, E+T) \overbrace{\Rightarrow}^{Tr1{,}2}$$
$$(q_1, i+i, T+T) \overbrace{\Rightarrow}^{Tr2{,}2} (q_1, i+i, F+T) \overbrace{\Rightarrow}^{Tr3{,}2} (q_1, i+i, i+T)$$
$$\overbrace{\Rightarrow}^{Tr6} (q_1, +i, +T) \overbrace{\Rightarrow}^{Tr7} (q_1, i, T) \overbrace{\Rightarrow}^{Tr2{,}2} (q_1, i, F)$$
$$\overbrace{\Rightarrow}^{Tr3{,}2} (q_1, i, i) \overbrace{\Rightarrow}^{Tr6} (q_1, \lambda, \lambda)$$

acepta

Limitaciones de los autómatas con pila y las GLC

Existen lenguajes que no son libre del contexto: no tienen gramáticas libres del contexto que los generen, ni autómatas con pila capaces de aceptarlos

$\Rightarrow$ **Los AP tienen limitaciones** a la hora de procesar cadenas:

- *La memoria de la pila es de acceso limitado.*
- *El recorrido por la cadena de entrada es limitado.*

Ejemplos de lenguajes que no son libres del contexto

- $L_{abc} = \{a^n b^n c^n \mid n \geq 0\}$

Limitaciones de los autómatas con pila y las GLC

Existen lenguajes que no son libre del contexto: no tienen gramáticas libres del contexto que los generen, ni autómatas con pila capaces de aceptarlos

$\rightrightarrows$ **Los AP tienen limitaciones** a la hora de procesar cadenas:

- *La memoria de la pila es de acceso limitado.*
- *El recorrido por la cadena de entrada es limitado.*

Ejemplos de lenguajes que no son libres del contexto

- $L_{abc} = \{a^n b^n c^n \mid n \geq 0\}$
- $L_{ww} = \{ww \mid w \in \{0,1\}^*\}$

Clasificación de gramáticas, lenguajes y máquinas

Noam Chomsky estableció una **clasificación de los lenguajes formales en función de los tipos de gramáticas capaces de describirlos:**

- **Lenguajes regulares (tipo 3).** Son los generados por **gramáticas regulares**. Conjunto $\mathcal{L}_{reg}$, también conocido como $\mathcal{L}_3$.
- **Lenguajes libres del contexto (tipo 2).** Son los generados por **gramáticas libre del contexto.** Conjunto $\mathcal{L}_{\mathsf{lc}}$ o $\mathcal{L}_2$.

Clasificación de gramáticas, lenguajes y máquinas

Noam Chomsky estableció una **clasificación de los lenguajes formales en función de los tipos de gramáticas capaces de describirlos:**

- **Lenguajes regulares (tipo 3).** Son los generados por **gramáticas regulares**. Conjunto $\mathcal{L}_{reg}$, también conocido como $\mathcal{L}_3$.
- **Lenguajes libres del contexto (tipo 2).** Son los generados por **gramáticas libre del contexto.** Conjunto $\mathcal{L}_{\mathsf{lc}}$ o $\mathcal{L}_2$.
- **Lenguajes sensibles al contexto (tipo 1).** Son los generados por **gramáticas sensibles al contexto**, con reglas de la forma:

  $$\boxed{\alpha A\beta \to \alpha\gamma\beta}$$

  La cadena γ no puede ser vacía, salvo en el caso $S \to \lambda$ y S no aparece en la parte derecha. Las cadenas α y β forman el **contexto** de la variable A y este contexto se mantiene al aplicar la regla. Conjunto $\mathcal{L}_{\mathsf{sc}}$ o $\mathcal{L}_1$.
  Ej.- Se pueden generar algunos lenguajes que no son libres del contexto, como $L_{abc} = \{a^n b^n c^n \mid n \geq 0\}$.

Clasificación de gramáticas, lenguajes y máquinas

Noam Chomsky estableció una **clasificación de los lenguajes formales en función de los tipos de gramáticas capaces de describirlos:**

- **Lenguajes regulares (tipo 3).** Son los generados por gramáticas regulares. Conjunto $\mathcal{L}_{reg}$, también conocido como $\mathcal{L}_3$.
- **Lenguajes libres del contexto (tipo 2).** Son los generados por **gramáticas libre del contexto.** Conjunto $\mathcal{L}_{\mathsf{lc}}$ o $\mathcal{L}_2$.
- **Lenguajes sensibles al contexto (tipo 1).** Son los generados por **gramáticas sensibles al contexto**, con reglas de la forma:

  $$\boxed{\alpha A \beta \to \alpha \gamma \beta}$$

  Conjunto $\mathcal{L}_{\mathsf{sc}}$ o $\mathcal{L}_1$.
- **Lenguajes recursivamente enumerables (tipo 0).** Son los generados por **gramáticas sin restricciones,** con reglas de la forma: $\boxed{\alpha \to \beta}$, donde en α aparece al menos una variable y β es cualquier cadena de símbolos de la gramática.

Clasificación de gramáticas y lenguajes

Teorema de jerarquí a de Chomsky Se cumple la siguiente **relación de inclusión propia entre conjuntos de lenguajes**:

$$\boxed{\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0}$$

Implicaciones:

- Si un lenguaje es de un tipo, podemos afirmar que también es un lenguaje que pertenece a una clase más amplia en la jerarquía.
- Si sabemos que un lenguaje pertenece a una clase no podemos excluir la posibilidad de que también sea un lenguaje de una clase más restringida.
  Ej.- si tenemos una gramática G que es libre del contexto no podemos afirmar de forma general que $L(G)$ no es regular, puede ser que exista G' equivalente y regular.

Complejidad del problema de validación (peor caso)

Cuanto más restringida sea la gramática más fácil será encontrar un algoritmo eficiente para **resolver el problema de la pertenencia** al lenguaje generado (**validación**).

Complejidad del problema de validación (peor caso)

Cuanto más restringida sea la gramática más fácil será encontrar un algoritmo eficiente para **resolver el problema de la pertenencia** al lenguaje generado (**validación**).

- **Lenguajes regulares:** $O(n)$. Se consigue mediante simulación del autómata finito determinista que acepta el lenguaje dado.

Complejidad del problema de validación (peor caso)

Cuanto más restringida sea la gramática más fácil será encontrar un algoritmo eficiente para **resolver el problema de la pertenencia** al lenguaje generado (**validación**).

- **Lenguajes regulares:** $O(n)$. Se consigue mediante simulación del autómata finito determinista que acepta el lenguaje dado.
- **Lenguajes libres del contexto:** $O(n^3)$. En lugar de simular un ap con backtracking en $O(2^n)$, se aplica un algoritmo llamado CYK, que se obtiene a partir de una glc, previa transformación a forma normal de Chomsky.

Complejidad del problema de validación (peor caso)

Cuanto más restringida sea la gramática más fácil será encontrar un algoritmo eficiente para **resolver el problema de la pertenencia** al lenguaje generado (**validación**).

- **Lenguajes regulares:** $O(n)$. Se consigue mediante simulación del autómata finito determinista que acepta el lenguaje dado.
- **Lenguajes libres del contexto:** $O(n^3)$. En lugar de simular un ap con backtracking en $O(2^n)$, se aplica un algoritmo llamado CYK, que se obtiene a partir de una glc, previa transformación a forma normal de Chomsky.
- **Lenguajes sensibles al contexto:** tiempo de ejecución exponencial.

Complejidad del problema de validación (peor caso)

Cuanto más restringida sea la gramática más fácil será encontrar un algoritmo eficiente para **resolver el problema de la pertenencia** al lenguaje generado (**validación**).

- **Lenguajes regulares:** $O(n)$. Se consigue mediante simulación del autómata finito determinista que acepta el lenguaje dado.
- **Lenguajes libres del contexto:** $O(n^3)$. En lugar de simular un ap con backtracking en $O(2^n)$, se aplica un algoritmo llamado CYK, que se obtiene a partir de una glc, previa transformación a forma normal de Chomsky.
- **Lenguajes sensibles al contexto:** tiempo de ejecución exponencial.
- **Lenguajes recursivamente enumerables:** tiempo de ejecución infinito, el problema de la validación no puede resolverse de forma algorítmica, es **indecidible**.

Jerarquía de lenguajes y máquinas