Universitas Onubensis · Sapere Aude ·
Universidad de Huelva

Tema 4

Autómatas de pila

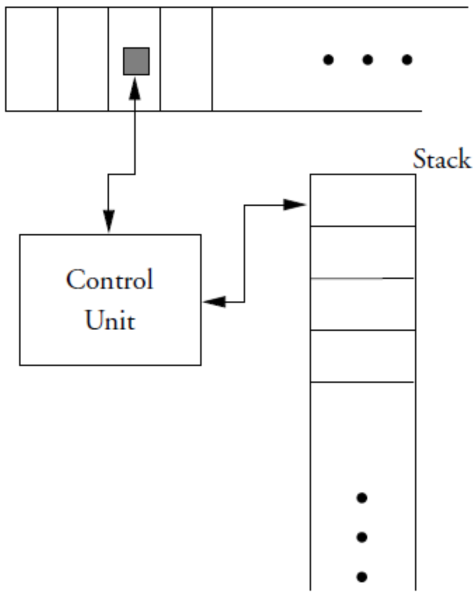
Índice

- Como hemos visto en la demostración del lema de bombeo para autómatas finitos, la limitación de los autómatas finitos se debe a que el número de estados en los que se puede encontrar el autómata está limitado.
- Para superar la capacidad de cómputo de los autómatas finitos debemos desarrollar un modelo que trabaje sobre una memoria ilimitada.
- En 1961, A.G. Oettinger propuso un modelo de autómata que trabaja sobre una memoria ilimitada en forma de pila. Las acciones que pueden desarrollarse sobre la memoria consisten en apilar o desapilar valores.

- Dado el lenguaje L = { 0^n 1^n | $n \geq 1$ }, no es posible construir un autómata finito que sea capaz de reconocerlo.
  - Intuitivamente, hace falta un estado para almacenar que se han leído un 0, dos 0s, tres 0s, … para poder comprobar después que el número de 1s es igual al número de 0s. Si el autómata finito tiene *m* estados, es imposible que pueda almacenar un estado que indique que el número de 0s es mayor que *m*.

- Sin embargo, es fácil reconocer este lenguaje por medio de una pila.
  - Si cada vez que se lee un 0 se almacena en la pila, el número de 0s que se pueden apilar es ilimitado. Para comprobar que el número de 1s es el mismo basta con ir desapilando un 0 por cada 1 que se lea en la cadena y aceptar si el último 1 corresponde al último 0 de la pila.

- Definición formal:
  - Un **autómata de pila** (**pushdown automaton – PDA**) se define como una sextupla $(\Sigma, \Gamma, Q, \Delta, q_0, F)$ donde:
    - Σ es el alfabeto de entrada, incluido el símbolo vacío β.
    - Γ es el alfabeto de la pila, incluido el símbolo vacío γ.
    - Q es el conjunto de estados del autómata.
    - Δ es el conjunto de transiciones
    - q_0 es el estado inicial del autómata
    - F es el conjunto de estados finales del autómata.
  - La unidad de control se describe como un Autómata Finito



- Transiciones del autómata
  - Las transiciones son del tipo:

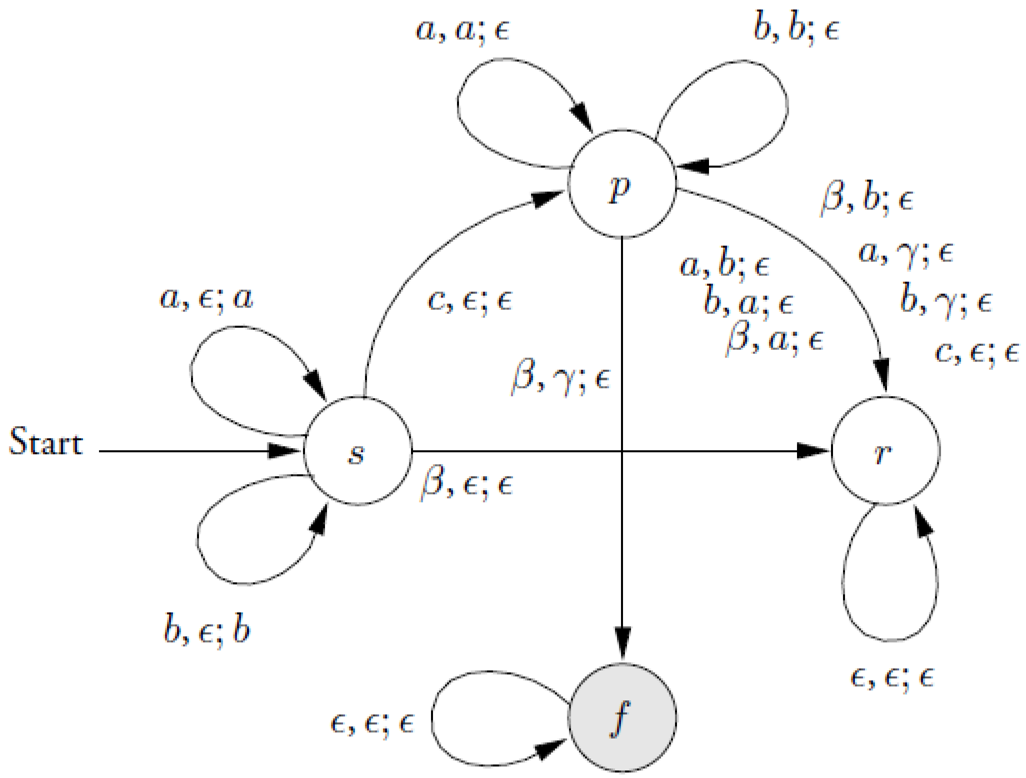
$$\Delta : \quad (Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times Q \times (\Gamma \cup \{\epsilon\}))$$

donde ϵ representa un símbolo vacio

  - Una transición $(p, x, y; q, z)$ significa que si el autómata se encuentra en el estado p, en la cinta de lectura se encuentra el símbolo x y en la cima de la pila se encuentra el símbolo y, se produce una transición hacia el estado q, se consume el símbolo x de la entrada, se desapila el símbolo y y se apila el símbolo z.
  - Si x= ϵ, entonces la transición no consume el símbolo de la entrada.
  - Si y= ϵ, entonces la transición no desapila el símbolo de la pila.
  - Si z= ϵ, entonces la transición no apila ningún símbolo en la pila.

- Funcionamiento del autómata de pila:
  - Inicialmente el autómata se encuentra en el estado q_0, la pila se encuentra vacía con el símbolo γ en el fondo y la cinta de lectura tiene la cadena de entrada w seguida de símbolos β.
  - El autómata va realizando transiciones, avanzando en la cinta de lectura y apilando y desapilando símbolos de la pila.
  - Existen dos modelos para definir como finalizar el proceso:
    1. La cadena de entrada se acepta si el estado del autómata tras consumir todos los símbolos de la cadena de entrada es un estado final.
    2. La cadena de entrada se acepta si la pila queda vacía al consumir todos los símbolos de la cadena de entrada.
  - Ambos modelos son equivalentes.

- Ejemplo de Autómata de Pila:
  - Autómata reconocedor del lenguaje $\{w\ c\ w^T\}$ donde $w \in (\{a,b\})^*$



- Autómatas de pila deterministas y no deterministas:
  - Un autómata de pila es ***determinista*** si en cada momento solo es posible realizar una transición. En caso contrario es un autómata ***no determinista***.
  - Formalmente, un autómata de pila es determinista si cumple las siguientes condiciones:
    1. Para cada $q \in Q$, $a \in \Sigma \cup \{ \epsilon \}$, $x \in \Gamma$, solo existe una o ninguna transición $\delta(q, a, x; p, z)$
    2. Para cada $q \in Q$, $x \in \Gamma$, si existe la transición $\delta(q, \epsilon, x; p, z)$ entonces no existen transiciones $\delta(q, a, x; p, z)$.

- Autómatas de pila deterministas y no deterministas:
  - En el caso de los Autómatas Finitos, los deterministas (AFD) y los no deterministas (AFND) tienen la misma capacidad de cómputo. El no determinismo significa que el AFND puede estar computando diferentes caminos simultaneamente. El "estado simultaneo" del AFND corresponde a un conjunto de estados de los diferentes caminos. Puesto que el número de estados es finito, el número de conjuntos de estados tambien lo es por lo que el comportamiento de un AFND con un conjunto de estados N puede simularse como un AFD con un conjunto de estados 2^N.

- Autómatas de pila deterministas y no deterministas:
  - En el caso de los Autómatas de Pila, los deterministas (APD) y los no deterministas (APND) no tienen la misma capacidad de cómputo.
  - El APND puede estar computando diferentes caminos simultaneamente.
  - En cada momento, la configuración del APND corresponde al estado de la unidad de control y al contenido de la pila.
  - En este caso es imposible simular el “estado simultaneo” (formado por varios estados y varias pilas) por medio de un único estado y una única pila. El problema está en que no es posible simular varias pilas con una única pila.
  - Por tanto, dado un APND es imposible obtener un APD equivalente.

- Autómatas de pila deterministas y no deterministas:
  - Por ejemplo, el lenguaje $\{ w\, w^T \}$ donde $w \in (\{a,b\})^*$ puede ser reconocido por un APND, pero no puede ser reconocido por un APD.
  - Demostración: el comportamiento del APND consiste en apilar w al reconocer la cadena y desapilarla al reconocer la cadena invertida. El problema es que no existe un símbolo que indique cuando termina w y comienza w^T. El APND puede probar simultaneamente todos los caminos, pero el APD no puede hacerlo.
  - Por ejemplo, la cadena "aa" puede ser una cadena completa (y en tal caso deberíamos apilar "a" y desapilarlo a continuación) o el comienzo de una cadena mayor (y en tal caso deberíamos apilar "aa"). Hasta que no se alcanza el final de la cadena es imposible saber en que punto era necesario comenzar a desapilar, pero el APD necesita tomar la decisión en el momento de leer cada carácter.

Lenguajes formales:

- Alfabeto (Σ): conjunto finito de símbolos
- Cadena: secuencia finita de símbolos
- Cadena vacía (λ): cadena de longitud cero
- Σ^k: Conjunto de cadenas de longitud k
- Clausura del alfabeto (Σ^*): Conjunto de todas las cadenas

$$\Sigma^* = \cup\, \Sigma^k = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$$

- Lenguaje formal: Subconjunto de Σ^*

Gramáticas formales:

- Forma de describir lenguajes formales
- Cuatro elementos (N, Σ, P, <S>):
- N: Alfabeto de símbolos no terminales
- Σ : Alfabeto de símbolos terminales
- P: Conjunto de producciones o reglas

$$P \subset (N \cup \Sigma)^* \, N \, (N \cup \Sigma)^* \rightarrow (N \cup \Sigma)^*$$

- <S>: Símbolo inicial (no terminal)

Lenguaje generado por una gramática:

- Derivación directa: η deriva directamente en γ ($\eta \Rightarrow \gamma$) si
  - $\eta = \omega_1\ \alpha\ \omega_2$
  - $\gamma = \omega_1\ \beta\ \omega_2$
  - $\alpha \rightarrow \beta\ \in P$
- Derivación: η deriva en γ ($\eta \overset{*}{\Rightarrow} \gamma$)si
  - existen $\xi_1,\ldots, \xi_n$ tales que η deriva directamente en ξ_1, ξ_i deriva directamente ξ_{i+1} y ξ_n deriva directamente en γ
- Forma sentencial: cadena $\alpha \in (N \cup \Sigma)^*$ tal que $<S> \overset{*}{\Rightarrow} \alpha$
- Sentencia: forma sentencial perteneciente a Σ^*
- Lenguaje generado por la gramática G:

$$L(G) = \{\ x \in \Sigma^* \mid <S> \overset{*}{\Rightarrow} x\ \}$$

- Jerarquía de Chomsky:

Nivel	Tipo de lenguaje	Dispositivo reconocedor
0	Con estrutura de frase	Máquina de Turing
1	Sensible al contexto	Autómata acotado linealmente
2	Libre de contexto	Autómata de Pila No Determinista
3	Regular	Autómata Finito

Gramáticas con estructura de frase:

- Son las que no presentan restricciones en cuanto a las reglas.
- Ejemplo de gramática con estructura de frase:

S → a S B C	a B → a b	c C → c c
S → a B C	b B → b b	
C B → B C	b C → b c	

- Ejemplo de derivación:

S → a S B C → a a S B C B C → a a a B C B C B C → a a a B B C C B C →
→ a a a B B C B C C → a a a B B B C C C → a a a b B B C C C → a a a b b B C C C →
→ a a a b b b C C C → a a a b b b c C C → a a a b b b c c C → a a a b b b c c c

- Esta gramática reconoce el lenguaje $L(G) = \{ a^n\, b^n\, c^n \mid n \geq 1\}$.

Gramáticas sensibles al contexto:

- Son aquellas en las que las reglas son de la forma (γ A $\beta \rightarrow \gamma$ B β). Es decir, podemos sustituir A por B siempre que se encuentre entre γ y β. Las cadenas γ y β representan el contexto en el que se puede aplicar la sustitucion de A.
- (Definición alternativa: gramáticas crecientes) Son aquellas en las que el número de símbolos en la parte derecha de las reglas es siempre mayor o igual que el de la parte izquierda. ($A \rightarrow B \Longrightarrow |A| \leq |B|$).
- Puede demostrarse que ambas definiciones conducen a gramáticas con la misma capacidad.
- Ejemplo de gramática sensible al contexto:

S → a S B C	C B → B C	b B → b b	c C → c c
S → a B C	a B → a b	b C → b c	

Gramáticas libres de contexto:

- Son aquellas cuyas reglas tienen la parte izquierda formada por un único símbolo no terminal. Por ejemplo, $A \rightarrow \alpha\, \beta\, \gamma$.
- Ejemplo de gramática libre de contexto:

$S \rightarrow a\, S\, b$
$S \rightarrow \epsilon$

- Ejemplo de derivación:

$$S \rightarrow a\, S\, b \rightarrow a\, a\, S\, b\, b \rightarrow a\, a\, a\, S\, b\, b\, b \rightarrow a\, a\, a\, b\, b\, b$$

- Esta gramática reconoce el lenguaje $L(G) = \{\, a^n\, b^n \mid n \geq 0 \,\}$

Gramáticas regulares:

- Son aquellas cuyas reglas en su parte derecha contiene un símbolo terminal o un símbolo terminal seguido de un no terminal.

$$A \rightarrow a \qquad\qquad A \rightarrow a\,B$$

- Ejemplo de gramática regular:

$S \rightarrow 0\,A$	$A \rightarrow 0\,S$	$B \rightarrow 1\,S$	$C \rightarrow 0\,B$	$A \rightarrow 0$
$S \rightarrow 1\,B$	$A \rightarrow 1\,C$	$B \rightarrow 0\,C$	$C \rightarrow 1\,A$	$B \rightarrow 1$

- Ejemplo de derivación:

$$S \rightarrow 0\,A \rightarrow 0\,1\,C \rightarrow 0\,1\,0\,B \rightarrow 0\,1\,0\,1$$

- Esta gramática reconoce un lenguaje formado por cadenas con un número par de 0s y de 1s.
- Las gramáticas regulares tienen la misma capacidad expresiva que las expresiones regulares y pueden ser reconocidas por Autómatas Finitos.

Forma Normal de Chomsky para gramáticas libres de contexto:

- Se dice que una gramática libre de contexto está en **Forma Normal de Chomsky** si todas sus reglas son de la forma $A \rightarrow a$ (parte derecha con un único símbolo terminal) o $A \rightarrow B\ C$ (parte derecha con dos símbolos no terminales). Si el lenguaje incluye la cadena vacía, entonces se incluye la regla $S \rightarrow \epsilon$ (siendo S el símbolo inicial).

- Ejemplo de gramática en Forma Normal de Chomsky:

$S \rightarrow A\ M$	$M \rightarrow S\ B$	$B \rightarrow b$
$S \rightarrow \epsilon$	$A \rightarrow a$	

Forma Normal de Chomsky para gramáticas libres de contexto:

- Toda gramática libre de contexto puede transformarse en una gramática en Forma Normal de Chomsky.
  - Las reglas A → ϵ se eliminan y para cada regla en la que aparezca el símbolo A se generan dos reglas, una con A y otra sin A.

S → s L M N M → a P Q M → ϵ	S → s L M N S → s L N M → a P Q

  - Para cada símbolo terminal *a* se crea un símbolo no terminal asociado *A* y una regla A → a. Cada ocurrencia de un símbolo terminal se sustituye por su símbolo no terminal asociado.
  - Las reglas A → B, donde B tiene asociadas reglas B → w, se sustituyen por reglas A → w, de forma que todas las reglas deben tener al menos dos símbolos no terminales
  - Cada regla (A → B C D ...) se sustituye por dos reglas (A → B X) y (X → C D …)

Algoritmo de reconocimiento para gramáticas libres de contexto:

- Dada una cadena de entrada w de tamaño n y una gramática G expresada en Forma Normal de Chomsky, ¿pertenece w al lenguaje descrito por G? ¿ $w \in L(G)$?
- Existen varios algoritmos que permiten resolver este problema de forma general: Cocke-Younger-Kasami, Early, GLR.
- Existen algoritmos más rápidos, pero están limitados a ciertos tipos de gramáticas.
- Algoritmo de Cocke-Younger-Kasami
  - Basado en programación dinámica. Es de orden $O(n^3 \cdot V)$, siendo n el tamaño de la entrada y V el número de símbolos no terminales de la gramática en Forma Normal de Chomsky.

Algoritmo de Cocke-Younger-Kasami:

- Se basa en el uso de una matriz $(n+1)\times(n+1)$ donde las celdas contienen conjuntos de símbolos no terminales.
- Paso 1: Cada celda $c_{i,i+1}$ contiene los símbolos no terminales que producen el símbolo terminal w_i.

$$c_{i,i+1} = \{ A \mid (A \rightarrow w_i) \in G\}$$

- Paso 2: Cada celda $c_{i,i+2}$ contiene los símbolos no terminales A que tienen reglas $A \rightarrow B\ C$ con $B \in c_{i,i+1}$ y $C \in c_{i+1,i+2}$.
- Paso j-ésimo: Cada celda $c_{i,j}$ contiene los símbolos no terminales A que tienen reglas $A \rightarrow B\ C$ con $B \in c_{i,k}$ y $C \in c_{k+1,j}$.

$$c_{i,j} = \{ A \mid (A \rightarrow BC) \in G,\ B \in c_{i,k} \text{ y } C \in c_{k+1,j} \}$$

- La cadena w se acepta si al repetir los pasos n veces, el símbolo inicial pertenece a la celda $c_{1,n}$.

APND asociado a una gramática libre de contexto:

- Dada una gramática libre de contexto G = (N, Σ, P, <S>) se puede construir un APND que reconozca el mismo lenguaje con la siguiente estructura:
  - El alfabeto de entrada es $\Sigma \cup \{ \beta \}$, siendo β el símbolo que marca el final de la entrada.
  - El alfabeto de la pila es $N \cup \Sigma \cup \{ \gamma \}$, siendo γ el símbolo que marca el fondo de la pila.
  - El estado inicial es *q*.
  - La primera transición es $(q, \epsilon, \epsilon; p, S)$, siendo *S* el símbolo inicial de la gramática.
  - Para cada símbolo $a \in \Sigma$, se incluye la transición $(p, a, a; p, \epsilon)$, que consume el símbolo *a* de la cadena de entrada y de la pila.
  - Para cada regla $A \rightarrow B\ C\ D$, siendo *B*, *C* y *D* símbolos terminales o no terminales, se incluyen las transiciones $(p, \epsilon, A; r1, D)$, $(r1, \epsilon, \epsilon; r2, C)$, $(r2, \epsilon, \epsilon; p, B)$, es decir, se desapila *A* y se apila la parte derecha de la regla en orden inverso (generando los estados *r* auxiliares que sean necesarios).
  - Se añade la transición final $(p, \beta, \gamma; f, \epsilon)$, siendo *f* el estado final del AFND.

Gramática libre de contexto asociada a un APND:

- Dado un autómata de pila $M = (\Sigma, \Gamma, Q, \Delta, s, F)$, que funciona como reconocedor de lenguaje y finaliza con la pila vacía, se puede generar una gramática libre de contexto G que genera el mismo lenguaje.
  - Los símbolos terminales de la gramática corresponden al alfabeto de entrada Σ, (sin el símbolo β).
  - Los símbolos no terminales de la gramática N, son el símbolo inicial S y los símbolos $<p, y, q>$, con $p \in Q$, $y \in \Gamma \cup \{ \epsilon \}$, $q \in Q$. Estos símbolos denotan el hecho de moverse del estado p al estado q en un conjunto de pasos con el único efecto sobre la pila de desapilar el símbolo y. Los estados $<p, \epsilon, q>$ reflejan el movimiento del estado p al q sin modificación de la pila. Los estados $<s, \epsilon, f>$, siendo s el estado inicial del APND y f un estado final, denotan la transición del estado inicial a un estado final en una serie de pasos, dejando la pila en su estado original. Esto debería suponer un proceso completo de reconocimiento de una cadena de entrada.

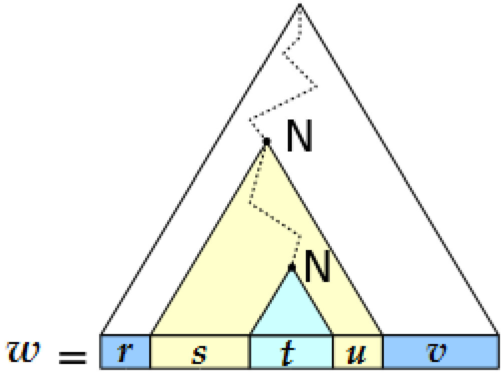
Gramática libre de contexto asociada a un APND:

- Dado un autómata de pila $M = (\Sigma, \Gamma, Q, \Delta, s, F)$, que funciona como reconocedor de lenguaje y finaliza con la pila vacía, se puede generar una gramática libre de contexto G que genera el mismo lenguaje.
  - Las reglas de la gramática son las siguientes:

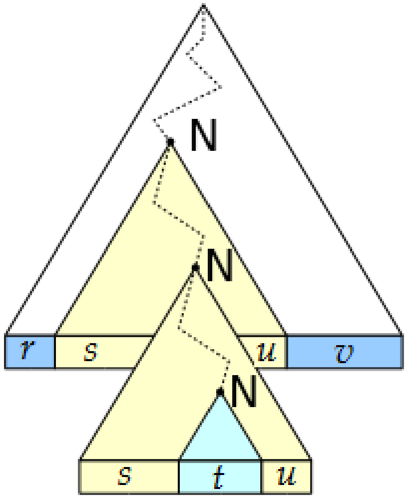
1. $S \rightarrow <s, \epsilon, f> \qquad \forall f \in F$
2. $<p, \epsilon, p> \rightarrow \epsilon \qquad \forall p \in Q$
3. $<p, y, r> \rightarrow x <q, z, r> \qquad \forall r \in Q, \ \forall <p,x,y; q, z> \in \Delta, \ y \neq \epsilon$
4. $<p, u, r> \rightarrow x <q,z,t> <t,u,r> \qquad \forall r,t \in Q, \ \forall <p,x, \epsilon; q, z> \in \Delta, \ \forall u \in \Gamma \cup \{ \epsilon \}$

- Si L es un lenguaje libre de contexto, existe un $p \geq 1$ tal que, toda cadena del lenguaje w de longitud mayor que p ($|w|>p$) puede escribirse como $w = r\ s\ t\ u\ v$, donde las subcadenas cumplen que:
  - $|\ s\ t\ u\ | \leq p$
  - $|\ s\ u\ | \geq 1$
  - $r\ s^n\ t\ u^n\ v \in \mathrm{L}$

- Demostración:
  - Puesto que la gramática libre de contexto tiene un número de símbolos no terminales finito, si una cadena w es lo suficientemente larga, entonces en su árbol de derivación debe repetirse algún símbolo no terminal N.

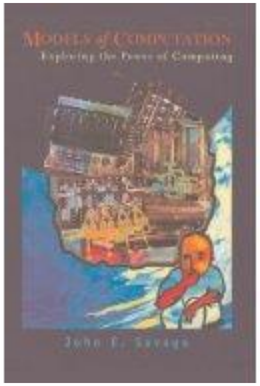


- Demostración:
  - Esto permite "bombear" la derivación de *N* a *N*



- El lema de bombeo para gramáticas libres de contexto identifica el tipo de lenguajes que se pueden reconocer por medio de autómatas de pila. Los lenguajes que no lo cumplan no pueden ser descritos por estos autómatas.
- Por ejemplo, los siguientes lenguajes no son libres de contexto:
  - $L = \{ a^n b^n c^n \mid n \geq 0 \}$
  - $L = \{ 0^n \mid n \text{ es primo} \}$
  - $L = \{ w\, w \mid w \in \{0,1\}^* \}$

Bibliografía



- Savage, John E. (1998). “Models Of Computation: Exploring the Power of Computing”. Capítulo 4