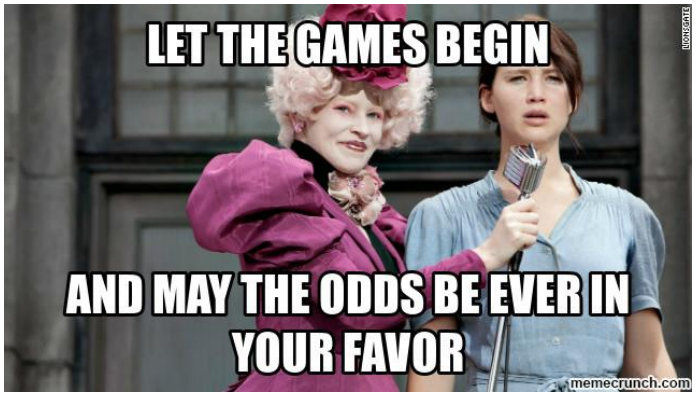
Compiladores e Intérpretes

Apuntes Clase
Viernes 10 de marzo-2017

Profesor:
Francisco Torres Rojas.

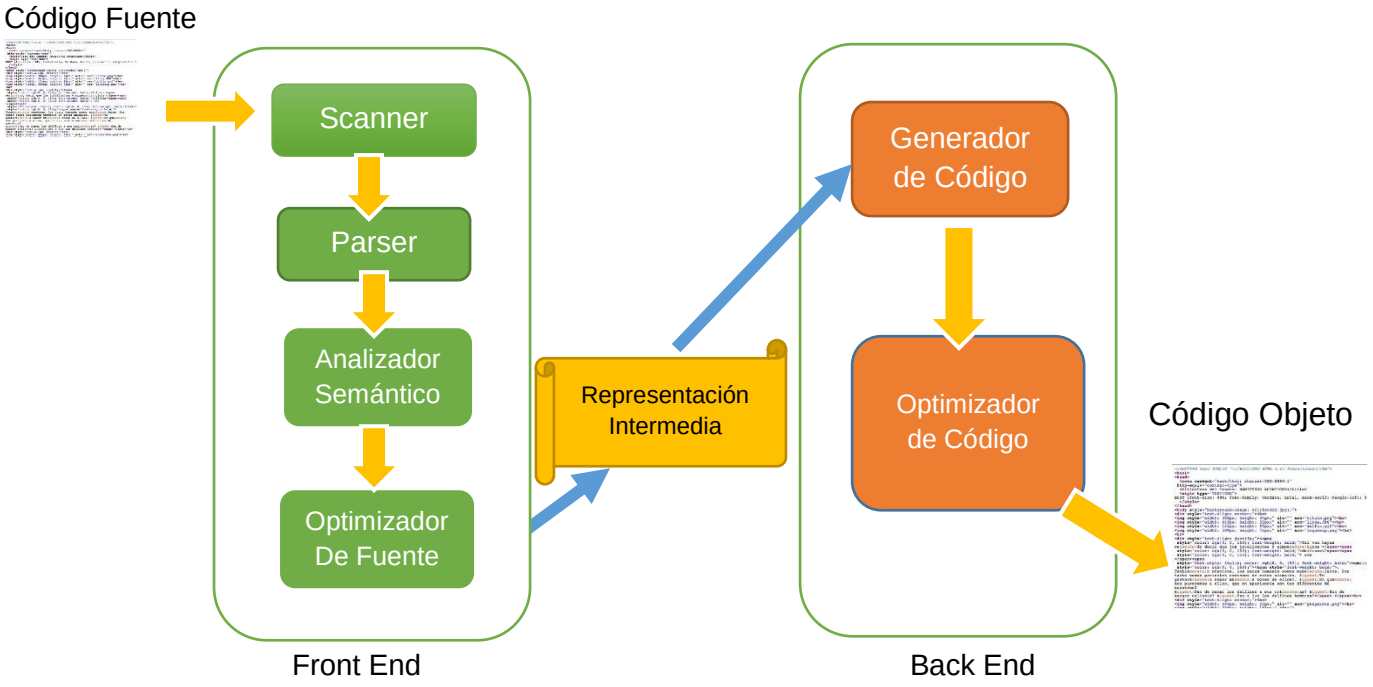
Grupo 40

Elaborado por:
Luis Diego Vargas Arroyo
2014036213



Análisis Léxico

Repasando lo que vimos la clase anterior.



Cada componente lo veremos en detalle en el trascurso de las clases.

Conceptos Generales

Hay algunos scanner, como este que no ocupan código de barras para leer. Como este que lee la cascara de la manzana y reconoce que es una.

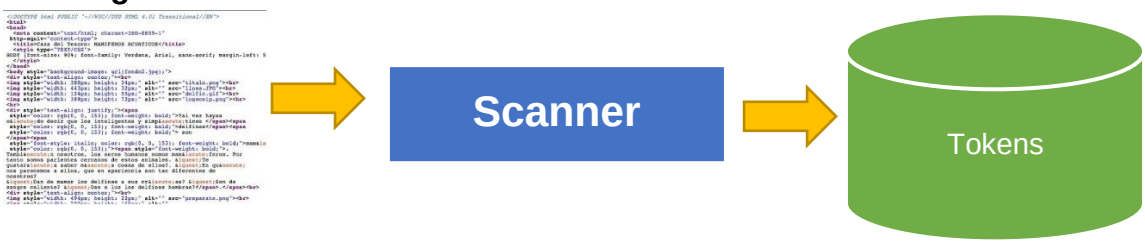
Oferta: Scribble es un lapicero que escanea el color de las cosas. El que se lo regale al profe es el **único que pasa el curso**. Vale $300 dejo el link de la página oficial https://www.scribblepen.com

Lo anterior fue para introducirnos a hablar sobre el scanner.

Scanner

Es el primero que se enfrenta al código fuente:

Código Fuente



Toma el código fuente de entrada y lo descompone categorías léxicas mínimas ósea en pedacitos más pequeños que tenga sentido.

Token: Un token puede equivale a una palabra en lenguaje natural, Usualmente es lo primero que se aprende en un lenguaje Natural.

Léxico = Vocabulario

Ejemplo:

- La primera palabra de un bebe es mamá. Ya que se lo repiten varias veces. Ellos escuchan un bla.. bla.. bla…. **Mamá**

O cuando oímos algo de **Scarlett Johansson,** Siempre nos detenemos cuando oímos o vemos algo de ella.

Detengámonos un segundo….

Continuemos….

Un ejemplo de lo que hace el scanner en el análisis léxico es el siguiente:

```
Burbuja ( N_entrada, vector[] )

Int índice, j, auxiliar;
        for (índice = 0; índice  <  N_entrada; índice++ )
                for(  j = índice+1  ; j < n  j++ )
                        if( vector[ j ] < vector [índice]){
                                aux= vector[j];
                                vector[j]=vector[indice]
                                vector[índice]= aux
                        }
```

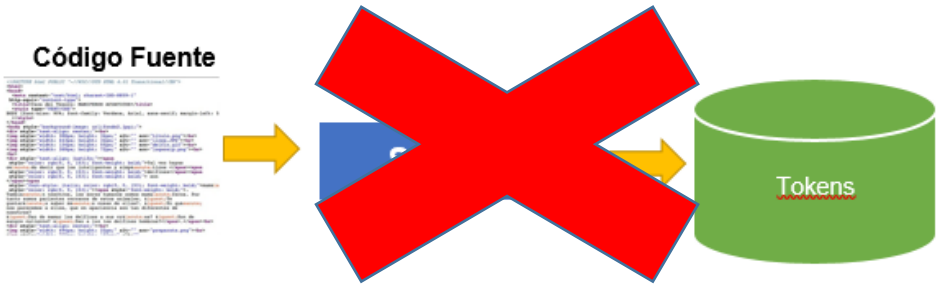
El scanner hace el esfuerzo de ir lo más largo posible hasta que tenga algo sentido. Por ejemplo:

- **Burbuja y N_entrada:** Tiene sentido debido a que puede ser el definidor de una función o de una variable
- **El paréntesis_izquierdo "(" y la coma ",":** También tiene sentido debido a que se usa como delimitadores.
- **Índice++ y j++:** Se podría decir que el scanner podría llegar solo a j+ o índice++ y tendría sentido, Pues sí. Pero el scanner también ve que j++ e índice++ tiene sentido y lo toma como un todo.

¿Qué pasa con los espacios?
Más adelante veremos que pasa con ellos

Versión Abstracta:

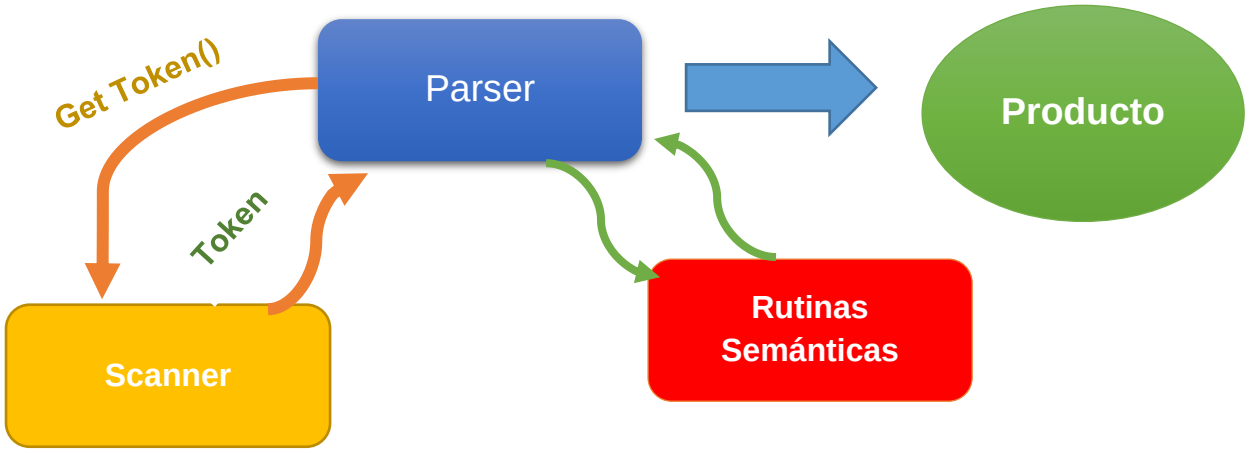
La representación que vimos anteriormente del scanner es incorrecta.



Traducción dirigida por sintaxis

Es la a organización más usual en los compiladores, la mayoría de compiladores modernos funcionan con esto.

- El parser dirige todo el proceso.
- El parser invoca al scanner cada vez que ocupa un token.
- Invoca rutinas semánticas en los puntos apropiados.
- El parser es el cliente del scanner.



Esta representación si es verdadera…. O tal vez no. Debido a que todo lo que nos dice el profe es mentira.

GetToken(): Es el encargado de regresar el siguiente Token.

Token

Un token representa un conjunto de hileras equivalentes sintácticamente(diferentes léxica y semántica).

Existen token de la siguiente manera:

- **Un solo elemento:** Ejemplo: El paréntesis_izquierdo **“ (”.**
- **Una cantidad pequeña de elementos:** Son lo que tienen como mucho 9 o 10 elementos. Ejemplos: **Float, Int, char, doublé.**
- **Conjunto potencialmente infinito:** Son los que poseen elementos infinitos de elementos. Ejemplo: **Números, hileras.**

Un token es una estructura con 2 campos:

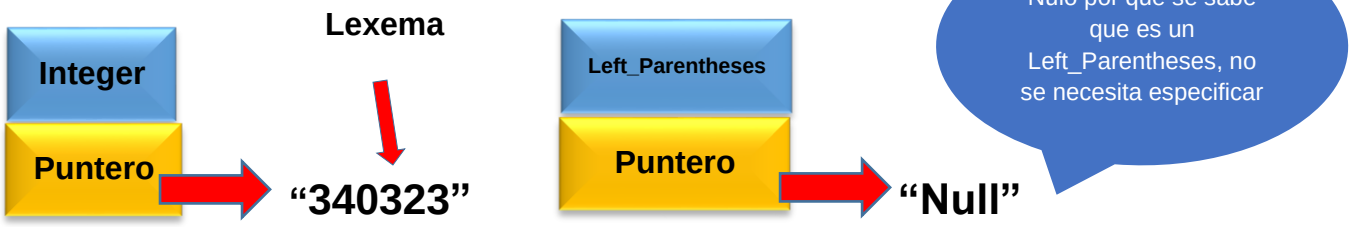
¿A dónde apunta este Puntero? Pues al lexema

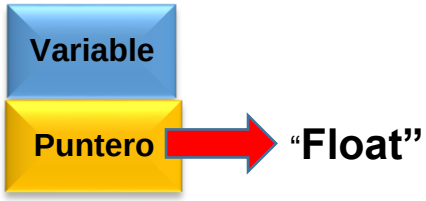
Un lexema es como el programador nombra a la variable.

Ejemplo: Una variable nombrada **cantidad_pollitos o izcar_1**

Token vrs Lexema

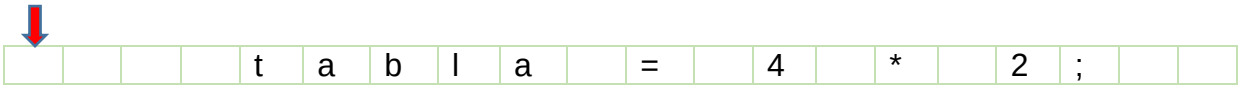
- Es una relación 1 a n Un token puede tener muchos lexama
- Un token es una categoría léxica
- Lexema es la hilera particular en el programa que corresponde al token reportado por el scanner.
- Ejemplos





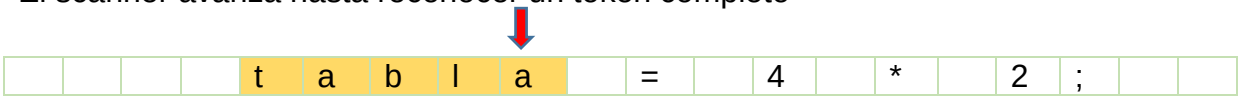
GetToken() en Ejecución

El scanner siempre mantiene su posición actual



-El scanner se brinca los “espacios”. Blancos, saltos de pagina, tabs, comentarios.

El scanner avanza hasta reconocer un token completo



La animación del profe está más chuza. ☺

- El scanner regresa el token.
- El scanner mantiene esa posición hasta que le soliciten otro token.

Scanner y “espacios”

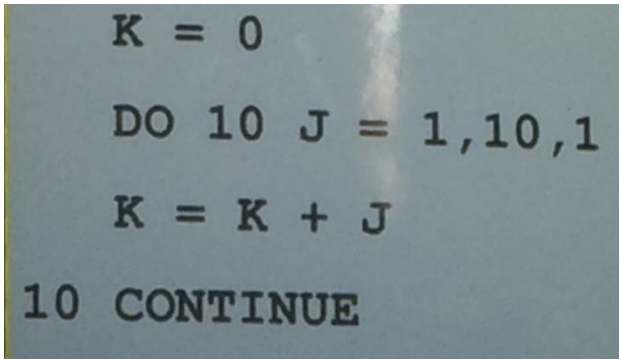
Ahora si hablaremos un poco de los espacios en los compiladores.

- Usualmente no hay un token que represente un **“espacio”.** Python puede que sí.
- Los espacios son muy importantes sintácticamente.

Idea de Backus con los espacios.

John Backus tuvo una duda filosófica mientras se encontraba en el baño. Y pensó **eliminemos todos los espacios** de los códigos de fortran y luego compilamos. **¿Buena idea?**

Ejemplo con el Do de fortran:



Se eliminan los espacios.

D	O	1	0	J	=	1	,	10	,	1

El scanner tiene que avanzar hasta reconocer un token.

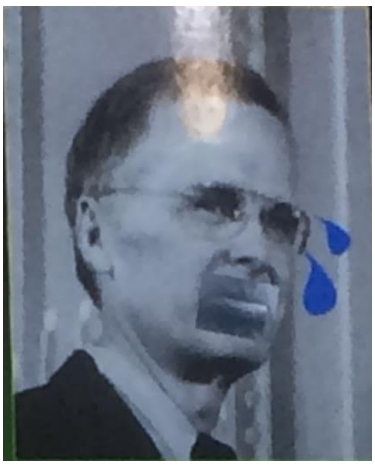
D	O	1	0	J	=	1	,	10	,	1			**D**	O	1	0	J	=	1	,	10	,	1

El scanner debido a que no hay espacios no sabe si es DO o un variable llamada DO10J y tiene una asignación de 1. Por lo cual sigue.

D	O	1	0	J	=	1	,	10	,	1

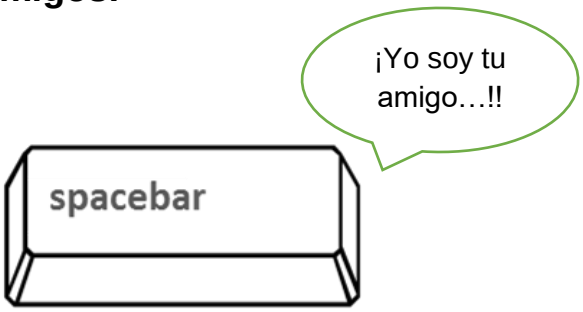
Y se da cuenta que era Un Do cuando llega a la coma, por lo tanto, se tiene que devolver y empezar de nuevo sabiendo que era un do.

Por lo cual no fue tan buena la de Backus de quitar los espacios.



Aunque los espacios no sean tokens, son buenos separadores.

Los espacios son nuestros amigos.



Generadores de Scanner

- Hay mucha teoría y experiencia acumulada respecto a Análisis léxico
- La mayoría se puede automatizar
- Generar un scanner es relativamente fácil.
- Siempre es bueno, aunque se utilice una herramienta, agregar un poco de código a mano.

Ejemplo de generadores de Scanner.

- Flex -Lex

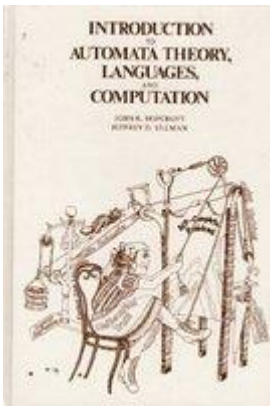
Para el proyecto 1 el profesor recomienda **Flex**

Detección de tokens

- Es la función principal que tiene que realizar el scanner
- Tiene que Avanzar en la entrada hasta reconocer el token más grande posible

¿Cómo se hace?

- Lenguaje formales
- Teoría de Autómatas
- Autómatas determinísticos de Estados Finitos



Biblia de Autómatas

Lenguajes Formales

Concéntrese en las definiciones. Que viene garrotazo.

Lenguaje: Habilidad humana para adquirir y usar complejos sistemas de comunicación. El estudio científico de lenguaje se llama **lingüística.**
Otra definición: Conjunto de sonidos articulados con que **el hombre** manifiesta lo que piensa o siente.
Ejemplos:
Lenguaje Natural: español, ingles, francés, italiano.

Lenguaje de Computadora: C, Java, ensamblador, Fortran.
Lenguaje matemáticos
Lenguajes formales

Símbolos

Es un concepto primitivo. Es la representación de un concepto, idea o identidad, con un significado convencional.

No necesariamente es lo mismo que un carácter.

- Nosotros denotaremos los símbolos como:

  a,b,c

Alfabeto

- Conjunto **finito** y no vacío de símbolos
- Usualmente denotado como $\sum$

Ejemplos:

$\sum$ = {A, B, C, D, ……, W, X, Y, Z}

$\sum$ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

$\sum$ = {0, 1}

$\sum$ = { A,T, C, G }

HILERA Sobre $\sum$

Es una secuencia de longitud arbitraria formada con símbolos tomados de un alfabeto $\sum$.

- Pueden repetirse
- El orden es muy importante
- También conocidos como palabras o frases
- Usualmente denotados como w, x, y, z

Ejemplos:

Sea $\sum$ = {0, 1} Posibles hileras serian:

- 10010101
-
- 1111
- 10101100110101

Sea $\sum$ ={A,T,C,G}

- ACAAATCG
- TGTTTAGCA
- TGAC

Longitud de una hilera

- Es la cantidad de símbolos de una hilera. La representación es la siguiente:

|w| es la longitud de la hilera w (cantidad de símbolos).

Ejemplos:

- |AAAATCATATATA|=13
- |00|=2
- |GUAGUAGUAGUAGUAGUAGUA|=21
- **La longitud puede ser = 0 Hilera vacía**

Para representar una hilera vacía se utiliza el épsilon o lambda: **ε**

| ε|=0

Prefijos de una Hilera

Sea W una hilera sobre ∑

Un prefijo de **w** es una hilera formada en el mismo orden de los primeros k símbolos desde la **"izquierda"** de **W.**

- **0 ≤ k ≤ |w|.**
- Si **k <|w|** tenemos un prefijo propio.
- **K = 0** tenemos el prefijo vacío.
- **ε** es prefijo de cualquier hilera.

Ejemplos.

Sea ∑={A,T,C,G}

Sea **w = TTGACACTGCAA** una hilera sobre ∑

-Algunos prefijos serian:

- TTGACA
- T
- TTGACACTGCAA
- TTGAC
- **ε**

Sufijos de una Hilera

Sea W una hilera sobre ∑

Un sufijo de **w** es una hilera formada en el mismo orden de los ultimos k símbolos desde la **"derecha"** de **w.**

- **0 ≤ k ≤ |w|.**
- Si **k <|w|** tenemos un sufijo propio.
- **K = 0** tenemos el sufijo vacío.
- ε es sufijo de cualquier hilera.

Ejemplos.

Sea ∑={A,T,C,G}

Sea **w = TTGACACTGCAA** una hilera sobre ∑

-Algunos sufijos serian:

- **GCAA**
- **A**
- **TTGACACTGCAA**
- **ACTGCAA**
- **ε**

Subhilera de una Hilera

Sea W una hilera sobre ∑

Una subhilera de **w** es una hilera formada tomando en el mismo orden k **símbolos consecutivos de w** a partir de una **posición J.**

- **0 ≤ k,j ≤ |w|.**
- **0≤ k+j ≤ |w|**
- **Si k+j < |w| y j>0** tenemos una subhilera propia.
- **K = 0** tenemos subhilera vacío.
- ε es subhilera de cualquier hilera.

Ejemplos:

Sea ∑={A,T,C,G}

Sea **w = TTGACACTGCAA** una hilera sobre ∑

Algunos **subhileras** serían:

- **GCAA**
- **A**
- **TTGACACTGCAA**
- **CACT**
- **CTGCA**
- **ε**

Concatenación de Hileras

Sean **v** y **w** dos hileras sobre $\sum$

- Si
  - **v= $a_1a_2a_3….a_k$**
  - **w= $b_1b_2b_3….b_n$**

La concatenación de v y w seria:

$a_1a_2a_3….a_kb_1b_2b_3….b_n$

Ejemplos:

Sea $\sum$ = {0,1}

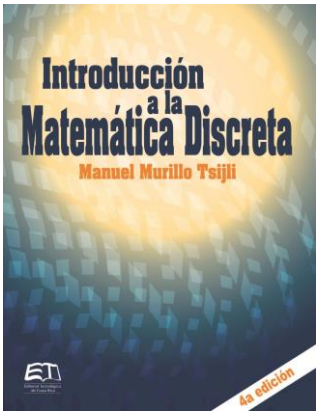
Consideremos las siguientes hileras:

- v = 111
- w = 01000
- x = 1101
- y = 011101110

Entonces:

- vw = 11101000
- xy = 1101011101110
- xv = 11011101
- wx = 010001101
- yy = 011101110011101110

Propiedades de la Concatenación de hileras



Asociativa

$$Vwx = (vw)x = v(wx)$$

En general, No conmutativa

$$Vw =/ wv$$

Longitudes se suman

$$|vw| = |v| + |w|$$

Neutro

La *hilera épsilon* es el elemento neutro de la concatenación

$$\mathbf{v\varepsilon = \varepsilon v = v}$$

Potencia n-ésima de una Hilera

Sea **v** una hilera sobre $\sum$

La Potencia n-ésima de **v** es la hilera resultado de concatenar **n** copias de **v**

v	v	v	v

Se denota como $\mathbf{v^n}$

Ejemplos:

Sea v = ATTAC

Entonces:

$\mathbf{V^2=}$ ATTACATTAC

$\mathbf{V^5=}$ ATTACATTACATTACATTACATTAC

$\mathbf{V^7=}$ ATTACATTACATTACATTACATTACATTACATTAC

Casos especiales

- $\mathbf{V^1= v}$
- $\mathbf{V^0= \varepsilon}$

Multiplicación de Alfabetos

Sea $\sum$ un alfabeto, entonces $\sum^{k}$ es el conjunto de todas las hileras sobre $\sum$ tales que tenga la longitud de **k.**

Sea $\sum = \{0,1\}$

- $\sum^{3}$ = {000,0001,010,011,100,101,110,110}
- $\sum^{2}$ = {00,10,01,11}
- $\sum^{0}$ es el conjunto que contiene a { ε }

Conjunto $\sum^{*}$

Sea $\sum$ un alfabeto, entonces $\sum^{*}$ se define recursivamente como:

1. ε pertenece a $\sum^{*}$
2. Si **w** pertenece $\sum^{*}$ y **a** pertenece $\sum$, entonces **wa** pertenece a $\sum^{*}$
3. W pertenece a $\sum^{*}$ solo si puede ser construida desde **ε** usando el paso 2 repetidamente

$$\sum^{*} = \bigcup_{i=0}^{\infty} \sum^{i}$$

Entonces $\sum^{*}$ es el conjunto de todas las hileras de cualquier longitud que se pueden formar a partir de símbolos tomados del alfabeto $\sum$

-Es un conjunto infinito

¿Quién fue el culpable de esto?

Stephen Cole Kleene

Fue un lógico y matemático estadounidense. Nació el 5 de enero del 1909 en Hartford, Connecticut, Estados Unidos y murió el 25 de enero de 1994.

- PHD en matemáticas- Princeton University
- Estudio con Alonzo Church
- Uno de los creadores de lenguajes formales
- Inventor de expresiones regulares
- Inventor del cierre de Kleene($\sum^*$)

También es actor de Hollywood. Conocido como Woody Harrelson

Ejemplo de $\sum^*$

Sea $\sum$= {0,1} entonces son miembros de $\sum^*$

- { ε , 0, 1, 00, 01,10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, 0001, 0010, 0011, 0100,0101, 0110, 0111, 1000,1001, 1010, 1011, 1100, 1110, 1111, 10000, …..}

- Entonces $\sum^+$ es el conjunto de todas las hileras de longitud mayor a 0 que se pueden formar a partir de símbolos tomados del alfabeto $\sum$
- Es un conjunto Infinito

Conjunto $\sum^+$

Sea $\sum$ un alfabeto, entonces $\sum^+$ se define recursivamente como:

1. **a** pertenece a $\sum^+$
2. Si **w** pertenece $\sum^+$ y a pertenece $\sum$, entonces **wa** pertenece a $\sum^+$
3. W pertenece a $\sum^+$ solo si puede ser construida desde algún elemento de $\sum$ usando el paso 2 repetidamente.

$$\sum^+ = \bigcup_{i=1}^{\infty} \sum$$

Ejemplo

Sea $\sum$= {0,1} entonces son miembros de $\sum^+$

- { 0, 1, 00, 01,10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, 0001, 0010, 0011, 0100,0101, 0110, 0111, 1000,1001, 1010, 1011, 1100, 1110, 1111, 10000, …..}

Esta fue toda la materia que se vio el día viernes 10 de marzo. No hubo garrote, pero fijo la siguiente clase sí.

Así estaremos todos en la siguiente clase.

Y recuerden que el miércoles 15 de marzo hay quiz.

Fin….