**Manual técnico**

**REQUERIMIENTOS TÉCNICOS:**

1. REQUERIMIENTOS MÍNIMOS DE HARDWARE

* Procesador: Intel (en cualquiera de sus procesadores), AMD (Cualquiera de sus procesadores).
* Memoria RAM: Mínimo: 1 Gigabytes (GB)
* Disco Duro: 500Gb.

1. REQUERIMIENTOS MÍNIMOS DE SOFTWARE

* Privilegios de administrador
* Sistema Operativo: Mínimo: Windows vista, 7, 8,10, en cualquiera de sus versiones.

**HERRAMIENTAS UTLIZADAS PARA SU DESARROLLO:**

Visual Estudio (2017):

**Microsoft Visual Studio** es un [entorno de desarrollo integrado](https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) (IDE, por sus siglas en inglés) para sistemas operativos [Windows](https://es.wikipedia.org/wiki/Windows). Soporta múltiples lenguajes de programación, tales como [C++](https://es.wikipedia.org/wiki/Visual_C%2B%2B), [C#](https://es.wikipedia.org/wiki/C_sharp), [Visual Basic .NET](https://es.wikipedia.org/wiki/Visual_Basic_.NET), [F#](https://es.wikipedia.org/wiki/F_Sharp), [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)), [Python](https://es.wikipedia.org/wiki/Python), [Ruby](https://es.wikipedia.org/wiki/Ruby) y [PHP](https://es.wikipedia.org/wiki/PHP), al igual que entornos de desarrollo web, como [ASP.NET](https://es.wikipedia.org/wiki/ASP.NET) MVC, Django, etc.

Visual Studio permite a los desarrolladores crear aplicaciones de escritorio, aplicaciones de sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas, entre otros.

**DESARROLLO:**

Creación del Autómata Finito Determinista:

**Método del Árbol:**

Es un método que se utiliza para crear un AFD (Autómata Finito Determinista) mínimo.

La técnica es una alternativa efectiva al método de Thomson ya que el método de Thomson no brinda el AFD mínimo.

* Partiendo de una expresión regular
* Permite la reducción de estados repetitivos.
* Produce autómatas más eficientes
* Permite la eliminación de nulos Creando un DFA óptimo.

Ya que la aplicación consiste en gran parte en un analizador léxico, para analizar el texto del código que se ingresa en un editor de texto se necesita definir un conjunto de símbolos o caracteres validos del lenguaje establecido:

**Para el desarrollo esta aplicación será válidos estos caracteres:**

N= { 1...9 } D={ 0...9 } T= todo menos “

S= { {, }, (, ), ‘,’, :, ; , [, ], 0, = } T1= todo menos \*

L= {A-Z a-z} T2=todo menos +

T3=todo menos /

Así mismo es una secuencia de [caracteres](https://es.wikipedia.org/wiki/Car%C3%A1cter_(tipo_de_dato)) que forma un patrón de búsqueda, principalmente utilizada para la [búsqueda de patrones](https://es.wikipedia.org/wiki/B%C3%BAsqueda_de_patrones) de cadenas de caracteres u operaciones de sustituciones. Por ejemplo, el grupo formado por las cadenas *Handel*, *[Händel](https://es.wikipedia.org/wiki/H%C3%A4ndel" \o "Händel)* y *Haendel* se describe con el patrón "H(a|ä|ae)ndel". Una expresión regular es una forma de representar los [lenguajes regulares](https://es.wikipedia.org/wiki/Lenguaje_regular) (finitos o infinitos) y se construye utilizando [caracteres](https://es.wikipedia.org/wiki/Tipograf%C3%ADa) del [alfabeto](https://es.wikipedia.org/wiki/Alfabeto), llamados Expresiones Regulares.

Toda expresión regular tiene algún autómata finito asociado. Específicamente, las expresiones regulares se construyen utilizando los operadores:

Alternación **‘|’**:

Una barra vertical separa las alternativas. Por ejemplo, "marrón|castaño" se corresponde con *marrón* o *castaño*.

Cuantificación:

Un cuantificador tras un carácter especifica la frecuencia con la que éste puede ocurrir. Los cuantificadores más comunes son "?", "+" y "\*":

**?:**

El signo de interrogación indica que el carácter que le precede puede aparecer como mucho una vez. Por ejemplo, "ob?scuro" se corresponde con *oscuro* y *obscuro*.

**+:**

El signo más indica que el carácter que le precede debe aparecer al menos una vez. Por ejemplo, "ho+la" describe el conjunto infinito *hola*, *hoola*, *hooola*, *hoooola*, etcétera.

**\*:**

El asterisco indica que el carácter que le precede puede aparecer cero, una, o más veces. Por ejemplo, "0\*42" se corresponde con *42*, *042*, *0042*, *00042*, etcétera.

Agrupación **()**:

Los paréntesis pueden usarse para definir el ámbito y precedencia de los demás operadores. Por ejemplo, "(p|m)adre" es lo mismo que "padre|madre", y "(des)?amor" se corresponde con *amor* y con *desamor*.

**Para el desarrollo del método la expresión regular a usar será:**

**ER: L(L|D|\_)\* | (-)?(ND\*(.D+)?|0.D+) | 0 |“T\*”| [‘\*’ T1\* ‘\*’] | / ‘\*’ T1\* ‘\*’/ | [‘+’T2\*‘+’] |S**

**Contrición del Árbol:**

Paso 1:

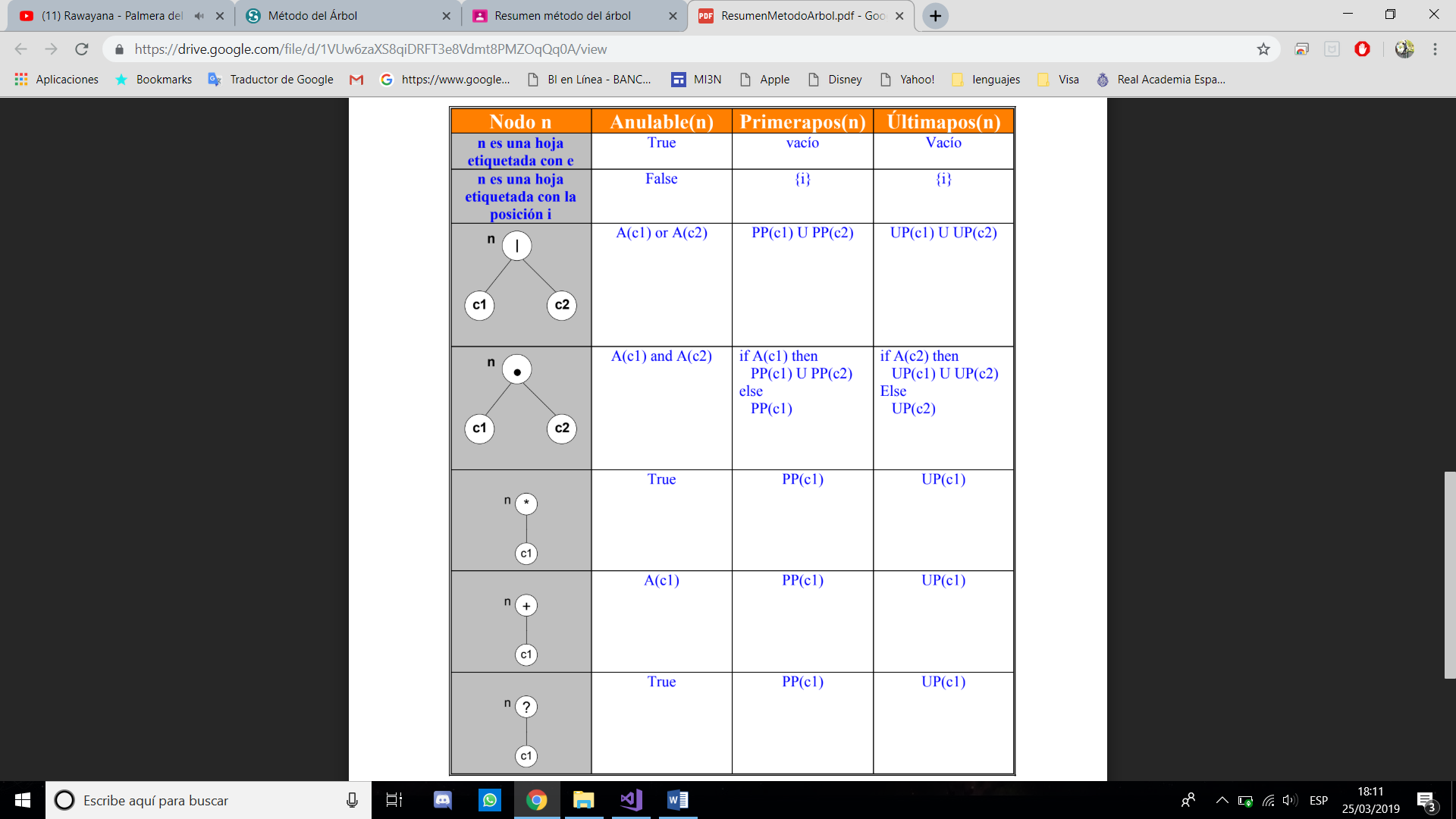
Aumentar la ER con #

**ER: ( L(L|D|\_)\* | (-)?(ND\*(.D+)?|0.D+) | 0 |“T\*”| [‘\*’ T1\* ‘\*’] | / ‘\*’ T1\* ‘\*’/ | [‘+’T2\*‘+’] |S)#**

Paso 2:

Crear el árbol asociado a la ER y enumerar las hojas

Donde todas las ultimas hojas del árbol son no Anulables F



Calcular Anulable, first, last, follow (primeros, últimos, siguientes):

SiguientePos(I): esta función indica qué posiciones pueden seguir a la posición I en el

árbol sintáctico.

Las reglas son:

1. Si n es un nodo de concatenación(.) con hijos izq c1 y der c2, e I es una posición

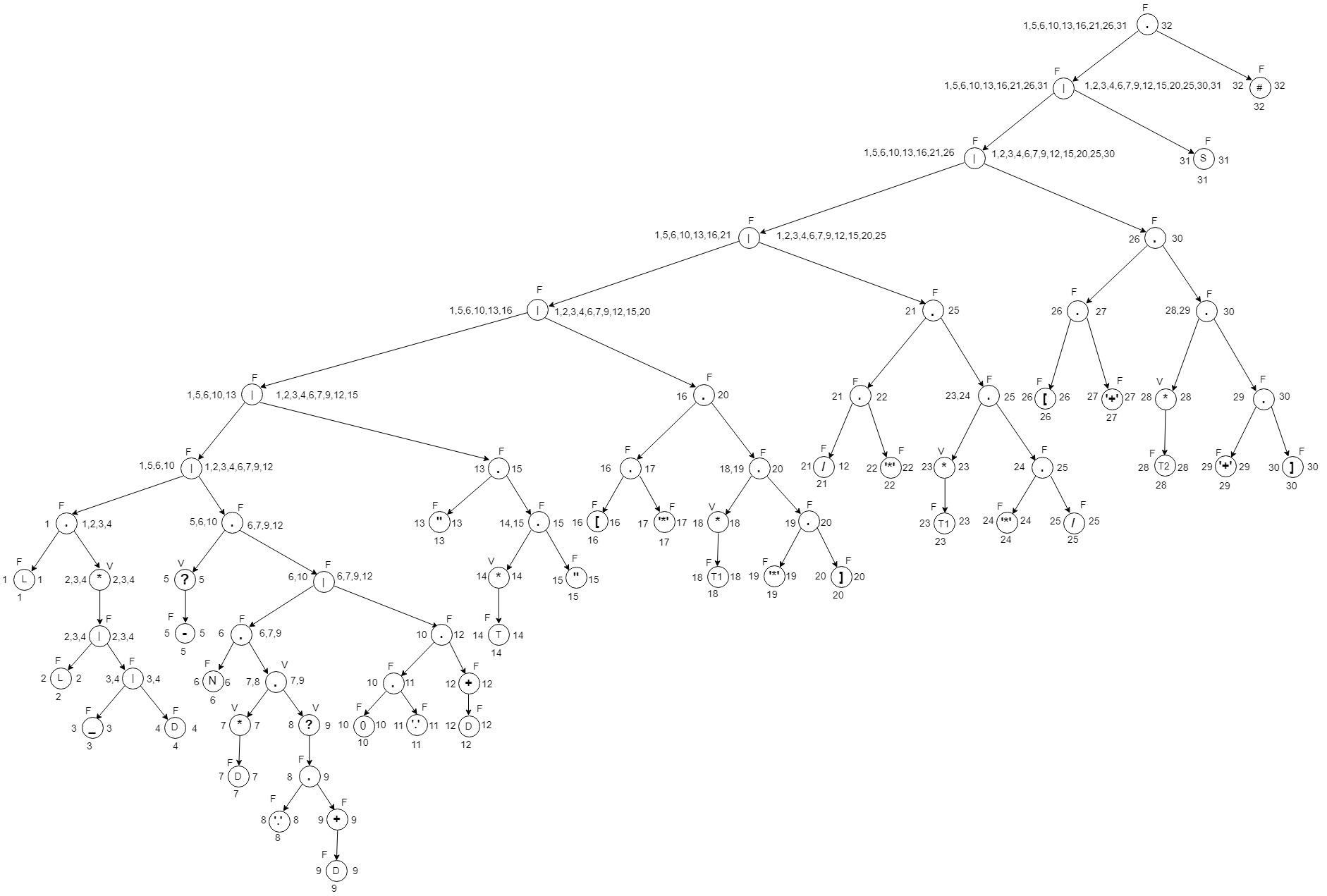
dentro de últimapos de c1, entonces todas las posiciones de primerapos de c2 están

en el siguientepos de i.

2. Si n es un nodo asterisco(\*) ó un nodo más (+), e I es una posición dentro de última

pos de n, entonces todas las posiciones de primerapos de n están en siguientepos de

I.



Construir el AFD.

