First and follow

Integrantes

- Mathew Cordero Aquino 22982
- Pedro Pablo Guzman 22111
- Diego Linares 221256

Estructura a seguir:

Link Repositorio

Video Youtube

grammar: Guarda las reglas como un mapa. Cada no terminal se asocia con una lista de producciones, donde cada producción es una lista de símbolos.

```
Ejemplo:
VP -> VP PP | V NP | eats
se guarda como:
"VP" → [ [ "VP", "PP" ], [ "V", "NP" ], [ "eats" ] ]
```

nonTerminals: Conjunto de símbolos no terminales (los que aparecen a la izquierda de ->).

terminals: Se infieren automáticamente. Son todos los símbolos que aparecen en las producciones y no son no terminales.

firstSet: Guarda los conjuntos FIRST ya calculados (memoización).

Algoritmo de First

La función computeFirst(symbol) funciona así:

```
Caso base:

    Si symbol es un terminal → FIRST(symbol) = { symbol }

    Si symbol = epsilon → FIRST(symbol) = { epsilon }

Caso recursivo (cuando symbol es un no terminal):

    Para cada producción de symbol, recorre los símbolos de izquierda a derecha.

    Agrega los símbolos de FIRST(X) al conjunto.

    Si todos los símbolos de una producción pueden derivar epsilon, entonces

    epsilon se agrega
```

al resultado.

Memoización:

Se guarda FIRST(symbol) ya calculado para evitar recalculaciones.

Algoritmo de follow

Reglas generales para calcular FOLLOW(A)

Regla 1: Si A es el símbolo inicial → FOLLOW(A) contiene $ (fin de

cadena).

Regla 2: Si hay una producción B -> alpha A Beta, entonces todo en FIRST

(Beta) (excepto epsilon) va a FOLLOW(A).

Regla 3: Si B -> alpha A o B -> alpha A Beta y FIRST(Beta) contiene

epsilon, entonces FOLLOW(B) va a FOLLOW(A).

Arquitectura final detallada:

explicación de la arquitectura y entrada

Para esta implementación en C++, desarrollamos un programa que calcula los conjuntos FIRST y FOLLOW de una gramática libre de contexto. Nuestra arquitectura sigue estos principios:

Input: El programa recibe una gramática a través de un archivo de texto

plano (gramatica.txt) donde cada línea representa una regla de producción

en el formato:

S -> NP VP VP -> VP PP | V NP | eats …

Parsing: Las reglas son leídas y almacenadas en una estructura tipo diccionario (map<string, vector<vector»), donde la clave es el no terminal y los valores son las diferentes producciones.

Identificación de terminales y no terminales: Los no terminales son los lados izquierdos de las producciones. Todo símbolo que aparece en el lado derecho pero no es no terminal, se considera terminal (excepto el símbolo especial epsilon).

FIRST: Se calcula de manera recursiva, siguiendo las reglas conocidas: si una producción empieza con un terminal, se agrega directamente; si empieza con un no terminal, se calcula su FIRST recursivamente. Si hay secuencias, se propagan correctamente y se considera epsilon.

FOLLOW: Se calcula de forma iterativa hasta que ya no haya cambios en los conjuntos. Se consideran las reglas estándar:

```
El símbolo inicial recibe $.

Si un símbolo A es seguido por una secuencia Beta, se agrega FIRST(Beta)



- epsilon a FOLLOW(A).

Si Beta puede derivar a epsilon, se agrega FOLLOW del símbolo de la

izquierda también.
```

Output: Se imprimen en consola los conjuntos FIRST y FOLLOW de cada no terminal.