Análisis sintáctico (Unidad 2)

En esta fase se necesita las gramáticas libres de contexto.

Una gramática libre de contexto (GlC) se componen de 4 cosas:

- Terminal Épsilon = {a, b}
- No terminal
- Axioma
- Reglas

C- usa:

- Descente
- LL(1)

LL(1): Derivación por la izquierda de un caracter a la vez: Derivación por la izquierda de un caracter a la vez.

```
void Start():{
    (D())+<EOF>;
}

void D():{
    T() L()
}

void T():{
    <Boolean> | <Char>
}

void L():{
    <Id> | Lp()
}

void Lp():{
    (<Coma> <Id> Lp())*
}
```

Eliminar recursión por la izquierda

$A \rightarrow A\alpha|\beta$

Reescribir estas reglas de tal manera que:

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A'|\epsilon$

Ejemplo (ver imagen):

Factorización por la izquierda

Ejemplo:

secuencia-sent → sent; secuencia-sent | sent

¿Cómo resolver?

$A \rightarrow \alpha\beta|\alpha\gamma$

Reescribir estas reglas como:

$A \rightarrow \alpha A'$

$A' \rightarrow \beta|\gamma$

Entrega de proyecto (2 semanas: 23 de septiembre)

Se entrega la siguiente parte del proyecto

---

lexp → atom | list

atom → numero | id

list → (lexp-seq)

lexp-seq → lexp-seq lexp | lexp

---

lexp → atom | list

atom → numero | id

list → (lexp-seq)

lexp-seq → lexp, lexp-seq | lexp

Tabla de símbolos

Todos los identificadores van a la tabla de símbolos

Operaciones de la tabla de símbolos**

- Búsqueda (lexema): entero;
- Inserción (lexema, descriptor): boolean;
- get_atributo (lexema, atributo): valor;
- Eliminación (lexema): entero;

Pila de tablas hashes para identificar scopes

Tabla de símbolos:

Id(lexema), Tipo de dato(int/real/void), Categoría(var/arreglo/función), Tamaño(del arreglo/número de parámetros)