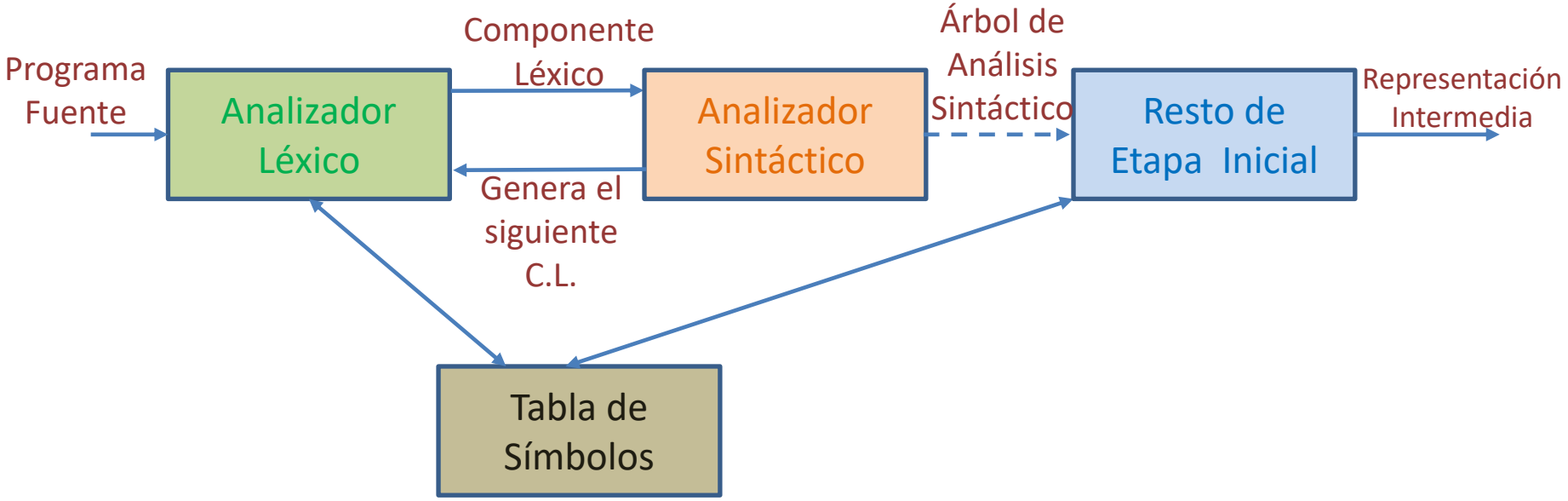
Análisis Sintáctico

Análisis Sintáctico



Analizador Sintáctico: Segunda fase de un compilador. Obtiene una cadena de componentes léxicos del Analizador Léxico, y verifica si la cadena puede ser generada por la gramática del lenguaje fuente. Informa de los errores sintácticos que se presenten.

Gramáticas Independientes del Contexto

Los lenguajes de programación tienen una estructura inherentemente recursiva que se puede definir mediante Gramáticas Independientes del Contexto (GIC).

Las GIC son un modelo matemático formado por una cuadrupla:

$$G = \{T, N, S, P\}$$

Gramáticas Independientes del Contexto

1. $\boldsymbol{T}$**: Terminales:** Símbolos básicos con los que se forman las cadenas. Componente léxico es sinónimo de terminal.
2. $\boldsymbol{N}$**: No terminales:** Son variables sintácticas que denotan conjuntos de cadenas. Estos definen conjuntos de cadenas que ayudan a definir el lenguaje generado por la GIC.
3. $\boldsymbol{S}$**: Símbolo Inicial:** Es un No terminal, y define a un conjunto de cadenas que representan al lenguaje generado por la GIC.
4. $\boldsymbol{P}$**: Producciones:** Especifican cómo se combinan los Terminales y los No Terminales para formar cadenas. Cada producción consta de dos partes:

Cabecera: Es un No Terminal, seguido por una flecha

Cuerpo: Cadena de Terminales y No Terminales.

Ejemplo: $A \rightarrow \alpha$ es una producción.

Gramáticas Independientes del Contexto

Ejemplo de GIC

Gramática de Operadores Aritméticos.

$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow E * E$

$E \rightarrow E/E$

$E \rightarrow (E)$

$E \rightarrow \boldsymbol{id}$

$T = \{+, -, *, /, (,), id\}$

$N = \{E\}$

$S = E$

P es el conjunto de producciones que aparece a la izquierda.

Escritura Abreviada

$E \rightarrow E + E | E - E |\ E * E | E/E | (E) | id$

Gramáticas Independientes del Contexto

Ejemplo de GIC

Gramática de Operadores Aritméticos.

$E \rightarrow E + T$

$E \rightarrow E - T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow T / F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow \boldsymbol{id}$

Escritura Abreviada

$E \rightarrow E + T \,|\, E - T \,|\, T$

$T \rightarrow T * F \,|\, T \,/F|\, F$

$F \rightarrow (E) \mid id$

$T = \{+, -, *, /, (,), id\}$

$N = \{E, T, F\}$

$S = E$

P es el conjunto de producciones que aparece a la izquierda.

Leer Convenciones de Notación. Pag. 171 Texto Guía

Gramáticas Independientes del Contexto

Ejemplo de GIC

$S \rightarrow (L) \mid \boldsymbol{a}$

$L \rightarrow L, S \mid S$

$T = \{(,), a, , \}$

$N = \{S, L\}$

$S = S$

P es el conjunto de producciones que aparece a la izquierda.

Gramáticas Independientes del Contexto

Derivaciones

Una GIC G define un lenguaje $L(G)$.

Las derivaciones establecen el proceso mediante el cual la GIC define un lenguaje.

Se considera cada producción como una regla de escritura, donde el No terminal a la izquierda es sustituido por la cadena del lado derecho de la producción.

Gramáticas Independientes del Contexto

Derivaciones

Ejemplo:

$$E \to E + E \,|\, E - E \,|\, E * E \,|\, E/E \,|\, (E) \,|\, id$$

Derivar la cadena: $id * id + id$

$$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow id * E + E$$
$$\Rightarrow id * id + E \Rightarrow id * id + id$$

Estas sustituciones se denominan *derivación* de la cadena $id * id + id$

Gramáticas Independientes del Contexto

Derivaciones

Reglas y Notas

1) Si $\alpha A \beta \Rightarrow \alpha \gamma \beta$, luego $A \rightarrow \gamma$ es una producción y α y β son cadenas arbitrarias de símbolos gramaticales.
2) Si $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$, se dice que α_1 *deriva* a α_n.
3) El símbolo $\Rightarrow$ significa *deriva en un paso*.
4) El símbolo $\overset{*}{\Rightarrow}$ significa *deriva en cero o más pasos*.

4.1) Si $\alpha \overset{*}{\Rightarrow} \alpha$, para cualquier cadena α.

4.2) Si $\alpha \overset{*}{\Rightarrow} \beta$ y $\beta \Rightarrow \gamma$, entonces $\alpha \overset{*}{\Rightarrow} \gamma$.

5) El símbolo $\overset{+}{\Rightarrow}$ significa *deriva en uno o más pasos*.

Gramáticas Independientes del Contexto

Derivaciones

Reglas y Notas

Dada una GIC G con símbolo inicial S, se utiliza la relación $\overset{*}{\Rightarrow}$ para definir $L(G)$, *el lenguaje generado por* G.

1. Las cadenas en $L(G)$ pueden contener sólo símbolos terminales de G; es decir, si ω está en $L(G)$, luego ω es generada por T^*.
2. ω está en $L(G)$ sí, y solo sí, $S \overset{+}{\Rightarrow} \omega$.

Gramáticas Independientes del Contexto

Derivaciones

Reglas y Notas

3. A la cadena ω se le llama *frase* de G.
4. A un lenguaje generado por una GIC, se le denomina *lenguaje independiente del contexto* (LIC).
5. Dos GIC que generan el mismo lenguaje, se les denomina GIC's equivalentes. Es decir, si $L(G) = L(G')$, luego $G \cong G'$.
6. Si $S \overset{*}{\Rightarrow} \alpha$, donde α puede contener no terminales, luego α es una *forma de frase* de G.
7. Una *frase* es una *forma de frase* sin no terminales.

Gramáticas Independientes del Contexto

Derivaciones

Ejemplo:

$E \rightarrow E + E | E - E |\ E * E | E / E | (E) | id$

En la derivación:

$$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow id * E + E \Rightarrow id * id + E$$
$$\Rightarrow id * id + id$$

$id * id + id$ es una *frase* de la GIC.

$E + E$, $E * E + E$, $id * E + E$, $id * id + E$, $id * id + id$ son formas de frases de la GIC.

Se escribe $E \overset{+}{\Rightarrow} id * id + id$ para indicar que $id * id + id$ se puede derivar de E.

Gramáticas Independientes del Contexto

Derivaciones

Tipos de derivaciones

$$E \rightarrow E + E \,|\, E - E \,|\, E * E \,|\, E/E \,|\, (E) \,|\, id$$

Derivación 1:

$$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow id * E + E \Rightarrow id * id + E \Rightarrow id * id + id$$

Derivación 2:

$$E \Rightarrow E * E \Rightarrow id * E \Rightarrow id * E + E \Rightarrow id * id + E \Rightarrow id * id + id$$

Gramáticas Independientes del Contexto

Derivaciones

Tipos de derivaciones

$$E \rightarrow E + E \,|\, E - E \,|\, E * E \,|\, E/E \,|\, (E) \,|\, id$$

Derivación 3:

$$E \Rightarrow E + E \Rightarrow E + id \Rightarrow E * E + id \Rightarrow E * id + id \Rightarrow id * id + id$$

Derivación 4:

$$E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow E * E + id \Rightarrow E * id + id \Rightarrow id * id + id$$

Gramáticas Independientes del Contexto

Derivaciones

Tipos de derivaciones

$E \rightarrow E + E | E - E |\ E * E | E / E | (E) | id$

Derivación 1: Más izquierda

$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow id * E + E \Rightarrow id * id + E \Rightarrow id * id + id$

Derivación 2: Más izquierda

$E \Rightarrow E * E \Rightarrow id * E \Rightarrow id * E + E \Rightarrow id * id + E \Rightarrow id * id + id$

Derivación 3: Más derecha

$E \Rightarrow E + E \Rightarrow E + id \Rightarrow E * E + id \Rightarrow E * id + id \Rightarrow id * id + id$

Derivación 4: Más derecha

$E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow E * E + id \Rightarrow E * id + id \Rightarrow id * id + id$

Gramáticas Independientes del Contexto

Derivaciones

Tipos de derivaciones

Derivación Más izquierda

Se sustituye el no terminal situado más a la izquierda de cualquier forma de frase.

$wA\gamma \underset{mi}{\Rightarrow} w\delta\gamma$, donde w está formada solo de terminales, $A \to \delta$ es la producción aplicada y γ es una cadena de símbolos gramaticales.

Aquí $w\delta\gamma$ es una *forma de frase izquierda* de la GIC.

Gramáticas Independientes del Contexto

Derivaciones

Tipos de derivaciones

Derivación Más derecha

Se sustituye el no terminal situado más a la derecha de cualquier forma de frase.

$\gamma Aw \underset{md}{\Longrightarrow} \gamma\delta w$, donde w está formada solo de terminales, $A \to \delta$ es la producción aplicada y γ es una cadena de símbolos gramaticales.

Aquí $\gamma\delta w$ es una *forma de frase derecha* de la GIC.

Gramáticas Independientes del Contexto

Derivaciones

Ambigüedad

Una GIC que genera una cadena con más de una derivación del mismo tipo, se considera ambigua.

Será ambigua la primera GIC de Operadores Aritméticos?

Será ambigua la segunda GIC de Operadores Aritméticos?

Gramáticas Independientes del Contexto

Derivaciones

Árbol de Análisis Sintáctico

Es una representación gráfica de una derivación que no muestra la elección relativa al orden de sustitución.

Cada nodo interior del AAS se etiqueta con algún no terminal A.

Los hijos de este nodo interior A, se etiquetan, de izquierda a derecha, con los símbolos del lado derecho de la producción por la cual se sustituyó esta A en la derivación.

Las hojas del AAS en cualquier momento se pueden leer de izquierda derecha y forman una forma de frase, llamada frontera del árbol.

Gramáticas Independientes del Contexto

Derivaciones

Árbol de Análisis Sintáctico

Ejemplo: Dada la GIC, construir los AAS correspondientes (Estar atentos al desplazamiento de la mano)

$E \rightarrow E + E | E - E |\ E * E | E / E | (E) | id$

Derivación 1: Más izquierda

$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow id * E + E \Rightarrow id * id + E \Rightarrow id * id + id$

Derivación 2: Más izquierda

$E \Rightarrow E * E \Rightarrow id * E \Rightarrow id * E + E \Rightarrow id * id + E \Rightarrow id * id + id$

Derivación 3: Más derecha

$E \Rightarrow E + E \Rightarrow E + id \Rightarrow E * E + id \Rightarrow E * id + id \Rightarrow id * id + id$

Derivación 4: Más derecha

$E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow E * E + id \Rightarrow E * id + id \Rightarrow id * id + id$

Gramáticas Independientes del Contexto

Derivaciones

Ejercicio: Tomar la segunda GIC de Aritmética de Operadores y resolver:

1. Es ambigua?
2. Construir el AAS para $id + id * id$ tanto para derivación más derecha como para izquierda.

Gramáticas Independientes del Contexto

Escritura de una Gramática

Expresiones regulares a GIC

Toda e.r. se puede escribir como una GIC. Las reglas a seguir son:

1) Construir un AF.
2) Para cada estado i del AF, crear un símbolo no terminal A_i.
3) Si $\delta(i, a) = j$, introducir $A_i \rightarrow aA_j$.
4) Si $\delta(i, \in) = j$, introducir $A_i \rightarrow A_j$.
5) Si i está en F, introducir $A_i \rightarrow \in$.
6) Si i es el estado de inicio del AF, luego A_i es el símbolo de inicio de la GIC.

Gramáticas Independientes del Contexto

Escritura de una Gramática

Expresiones regulares a GIC

Ejemplo:

Convertir a una GIC la e.r. $(a|b)^*abb$.

A partir del AFD óptimo:

$$A \rightarrow aB|bA$$
$$B \rightarrow aB|bC$$
$$C \rightarrow aB|bD$$
$$D \rightarrow aB|bA|\in$$

Gramáticas Independientes del Contexto

Escritura de una Gramática

Eliminación de Recursividad por Izquierda

Una GIC es *recursiva por izquierda* si tiene un no terminal A tal que existe una derivación $A \overset{+}{\Rightarrow} A\alpha$ para alguna cadena α.

Los analizadores sintácticos descendentes no pueden manejar gramáticas recursivas por la izquierda.

Gramáticas Independientes del Contexto

Escritura de una Gramática

Eliminación de Recursividad por Izquierda

Método Simple:

Dada la GIC $G: A \rightarrow A\alpha|\beta$, la GIC G' sin recursividad:

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A'| \in$

Ejemplo: Dada la GIC, eliminar la recursividad

$S \rightarrow (L) \mid \boldsymbol{a}$

$L \rightarrow L, S \mid S$

Solución:

$S \rightarrow (L) \mid \boldsymbol{a}$

$L \rightarrow SL'$

$L' \rightarrow, SL'| \in$

Gramáticas Independientes del Contexto

Escritura de una Gramática

Eliminación de Recursividad por Izquierda

Método Compuesto:

Dada la GIC G: $A \rightarrow A\alpha_1|A\alpha_2|\ldots|A\alpha_n|\beta_1|\beta_2|\ldots|\beta_m$, la GIC G' sin recursividad es:

$A \rightarrow A(\alpha_1|\alpha_2|\ldots|\alpha_n)|(\beta_1|\beta_2|\ldots|\beta_m)$

Si $\alpha = \alpha_1|\alpha_2|\ldots|\alpha_n$ y $\beta = \beta_1|\beta_2|\ldots|\beta_m$, luego

$A \rightarrow A\alpha|\ \beta$, se convierte en el caso simple y se elimina igual:

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A'|\ \in$

Gramáticas Independientes del Contexto

Escritura de una Gramática

Eliminación de Recursividad por Izquierda

Método Compuesto:

$A \rightarrow A\alpha| \beta$, se convierte en el caso simple y se elimina igual:

$A \rightarrow (\beta_1|\beta_2| \ldots |\beta_m)A'$

$A' \rightarrow (\alpha_1|\alpha_2| \ldots |\alpha_n)A'| \in$

Expandiendo la expresión:

$A \rightarrow \beta_1 A'|\beta_2 A'| \ldots |\beta_m A'$

$A' \rightarrow \alpha_1 A'|\alpha_2 A'| \ldots |\alpha_n A'| \in$

Gramáticas Independientes del Contexto

Escritura de una Gramática

Eliminación de Recursividad por Izquierda

Método Compuesto:

Ejemplo: Dada la GIC, eliminar la recursividad

$E \rightarrow E + T \mid E - T \mid T$

$T \rightarrow T * F \mid T \ / F \mid F$

$F \rightarrow (E) \mid id$

Solución:

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid -TE' \mid \in$

$T \rightarrow FT'$

$T' \rightarrow * FT' \mid /FT' \mid \in$

$F \rightarrow (E) \mid id$

Gramáticas Independientes del Contexto

Escritura de una Gramática

Factorización por Izquierda

Otra transformación importante de una GIC para utilizarla en Analizador Sintáctico Descendente.

Dada la GIC $A \rightarrow \alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n|\gamma$,

Método:

Se transforma en la GIC equivalente:

$A \rightarrow \alpha A'|\gamma$

$A' \rightarrow \beta_1|\beta_2|\ldots|\beta_n$

Gramáticas Independientes del Contexto

Escritura de una Gramática

Factorización por Izquierda

Ejemplo: Dada la GIC, factorizar a izquierda

$P \rightarrow iEtP|iEtPeP|a$

$E \rightarrow b$

Solución:

$P \rightarrow iEtPP'|a$

$P' \rightarrow \in |eP$

$E \rightarrow b$

Gramáticas Independientes del Contexto

Escritura de una Gramática

Construcción de Lenguajes Independientes del Contexto

Construir la GIC a partir de los siguientes lenguajes expresados en forma comprensiva:

1) $L_0 = \{wcw | w \text{ está en } (a|b)^*\}$
2) $L_1 = \{wcw^I | w \text{ está en } (a|b)^*\}$
3) $L_2 = \{a^n b^n | n \geq 1\}$
4) $L_3 = \{a^n b^m c^m d^n | n, m \geq 1\}$
5) $L_4 = \{a^n b^n c^m d^m | n, m \geq 1\}$
6) $L_5 = \{a^i b^j c^k | j, k \geq 1, i = j + k\}$
7) $L_6 = \{a^i b^j c^k | i, k \geq 1, j = i + k\}$
8) $L_7 = \{a^i b^j c^k | i, j \geq 1, k = i + j\}$

Análisis Sintáctico Descendente

Análisis sintáctico por descenso recursivo

1. Este análisis se considera un intento de encontrar una derivación por la izquierda para una cadena de entrada.
2. Se considera también un intento por construir un árbol de análisis sintáctico para la entrada comenzando desde la raíz y creando los nodos en pre-orden.

Análisis Sintáctico Descendente

Análisis sintáctico por descenso recursivo

Con retroceso

Dada la GIC:

$S \rightarrow cAd$

$A \rightarrow ab|a$

Reconocer la cadena $w = cad$

Análisis Sintáctico Descendente

Análisis sintáctico por descenso recursivo

Con retroceso

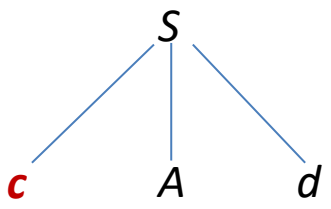
$S \rightarrow cAd$

$A \rightarrow ab|a$

Reconocer la cadena $w = cad$

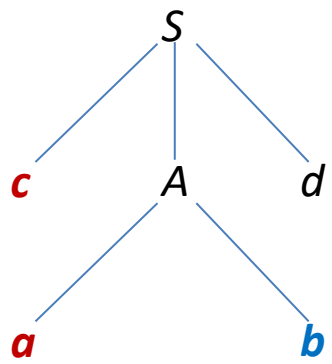
$w = \boldsymbol{c}ad$

Se crea el árbol inicial con un solo nodo ***S***



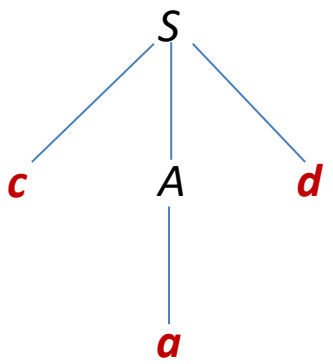
$w = c\boldsymbol{a}d$

Se expande el árbol con la primera producción de ***S.***
Se empareja la hoja ***c*** con ***c***ad



$w = ca\boldsymbol{d}$

No hay coincidencia con el siguiente símbolo de entrada.
No empareja con ***d***



$w = ca\boldsymbol{d}$

Se retrocede para probar con la siguiente producción de A.
Si empareja con ***d***

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Objetivo: Construir un Análisis Sintáctico Descendente Predictivo No Recursivo.

Método: Se usa una pila en vez de llamados recursivos. El principal problema es conocer qué producción debe aplicarse a un no terminal.

Salida: Reconocimiento de una cadena *w* como parte de un LIC generado por una GIC.

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Funciones **PRIMERO** y **SIGUIENTE**

PRIMERO(α): Conjunto de símbolos terminales que inician las cadenas derivadas de α. Si $\alpha \overset{*}{\Rightarrow} \in$, entonces $\in$ también está en PRIMERO(α).

Ejemplo:

Tomar alguna de las GIC desarrolladas y probar el PRIMERO.

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Funciones **PRIMERO** y **SIGUIENTE**

SIGUIENTE(A): Conjunto de símbolos terminales que pueden aparecer a la derecha de A en alguna forma de frase.

Si $S \overset{*}{\Rightarrow} \alpha A a \beta$, para algún α y β, el terminal a está en SIGUIENTE(A).

Si en algún momento de la derivación hubo símbolos entre A y a, derivaron $\in$, y desaparecieron. $\in$ no está en SIGUIENTE(A).

Si A está más a la derecha en una forma de frase, se agrega \$ a SIGUIENTE(A).

Ejemplo:

Tomar alguna de las GIC desarrolladas y probar el SIGUIENTE.

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Reglas para calcular el **PRIMERO**

Para calcular PRIMERO(X), para todos los símbolos gramaticales X, aplíquense las siguientes reglas hasta que no se pueda añadir más terminales o $\in$ a ningún conjunto PRIMERO

1. Si X es terminal, PRIMERO(X) es $\{X\}$.
2. Si $X \rightarrow \in$, entonces añadir $\in$ a PRIMERO(X).

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Reglas para calcular el **PRIMERO**

3. Si X es no terminal y $X \rightarrow Y_1 Y_2 \ldots Y_k$ es una producción, entonces añadir a en PRIMERO(X) si a está en PRIMERO(Y_i) y $\in$ en todos los PRIMERO(Y_1),…, PRIMERO(Y_{i-1}), para alguna i.

   Es decir, si $Y_1 Y_2 \ldots Y_{i-1} \overset{*}{\Rightarrow} \in$ y a está en PRIMERO(Y_i), se agrega a a PRIMERO(X).

   Si $\in$ está en PRIMERO(Y_j), para toda $j = 1,2, \ldots, k$, entonces se añade $\in$ a PRIMERO(X).

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Reglas para calcular el **PRIMERO**

Ejemplo:

Para la GIC de operadores, calcular el PRIMERO de cada símbolo no terminal.

Gramática de Operadores Recursiva a Izquierda

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$

Gramática Sin Recursividad

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \in$

$T \rightarrow FT'$

$T' \rightarrow * FT' \mid \in$

$F \rightarrow (E) \mid id$

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Reglas para calcular el **SIGUIENTE**

Para calcular SIGUIENTE(A), para todos los símbolos no terminales A, aplíquense las siguientes reglas hasta que no se pueda añadir más terminales a ningún conjunto SIGUIENTE.

1. Añadir \$ a SIGUIENTE(S), donde S es el símbolo inicial y \$ es el delimitador derecho de la entrada.
2. Si $A \rightarrow \alpha B \beta$, entonces PRIMERO($\beta$) excepto $\in$ se añade en SIGUIENTE(B).
3. Si $A \underset{*}{\Rightarrow} \alpha B$ o $A \rightarrow \alpha B \beta$, donde PRIMERO($\beta$) contiene $\in$ (o $\beta \Rightarrow \in$), entonces todo SIGUIENTE(A) está en SIGUIENTE(B).

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Reglas para calcular el **SIGUIENTE**

Ejemplo:

Para la GIC de operadores, calcular el SIGUIENTE de cada símbolo no terminal.

Gramática de Operadores Recursiva a Izquierda

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$

Gramática Sin Recursividad

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \in$

$T \rightarrow FT'$

$T' \rightarrow * FT' \mid \in$

$F \rightarrow (E) \mid id$

Análisis Sintáctico Descendente

Análisis sintáctico predictivo no recursivo

Cálculo de PRIMERO y SIGUIENTE

Ejemplo:

Para la GIC de operadores, calcular el SIGUIENTE de cada símbolo no terminal.

Gramática Sin Recursividad

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \in$

$T \rightarrow FT'$

$T' \rightarrow * FT' \mid \in$

$F \rightarrow (E) \mid id$

PRIMEROS:

Prim(E)={id,(}

Prim(E')={+,∈}

Prim(T)={id,(}

Prim(T')={*,∈}

Prim(F)={id,(}

SIGUIENTES:

Sgte(E)={$,)}

Sgte(E')={$,)}

Sgte(T)={+,$,)}

Sgte(T')={+,$,)}

Sgte(F)={*,+,$,)}