



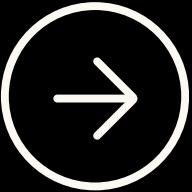
FIUSAC

ANÁLISIS SINTÁCTICO

Josue Zuleta



AGENDA

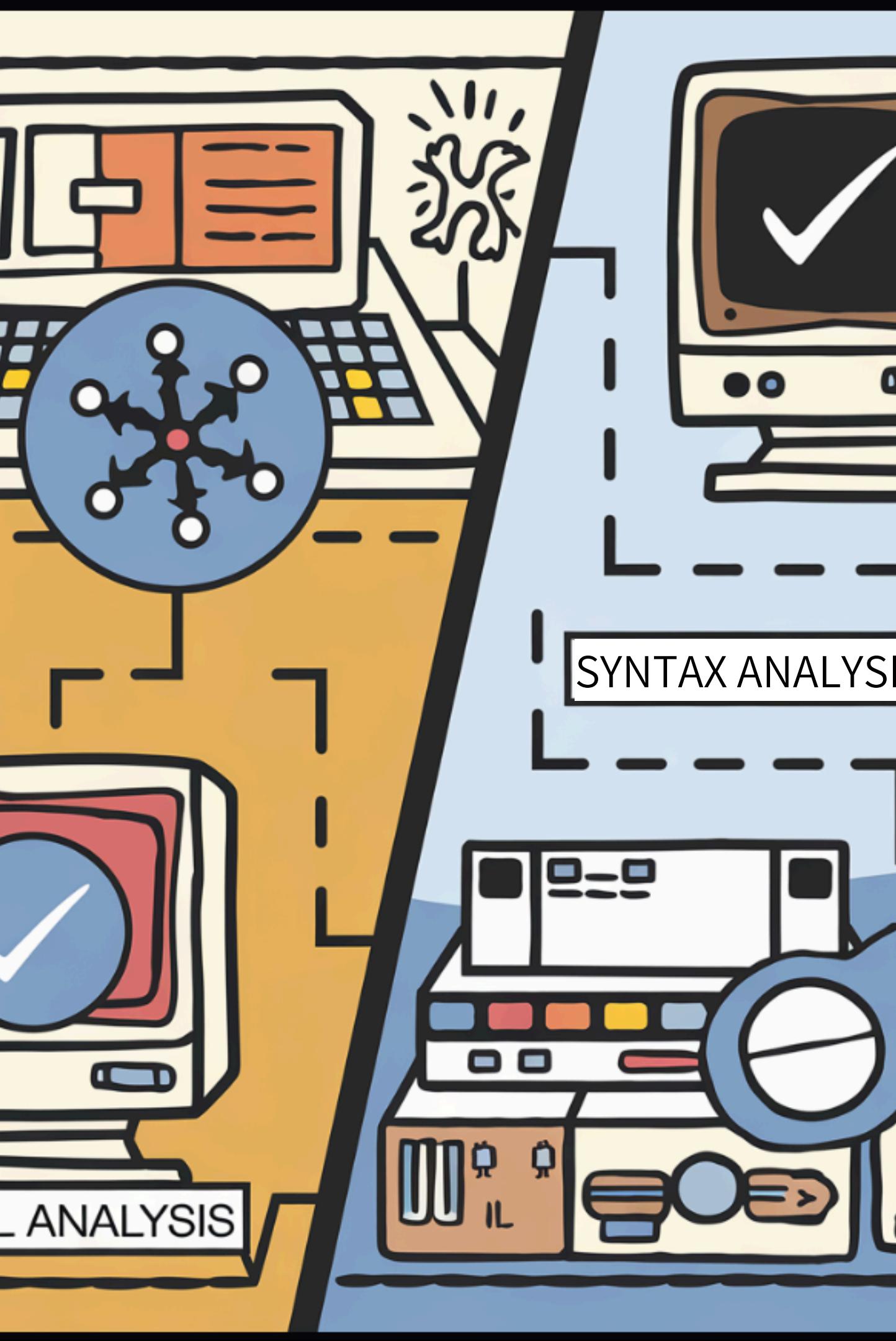


- 1 Sintaxis
- 2 Gramática
- 3 Gramática tipo 2
- 4 Ejemplos de gramáticas
- 5 Parser
- 6 CST vs AST
- 7 Tabla de símbolos
- 8 Tratamiento de errores
- 9 Ejemplo práctico

INTRODUCCIÓN

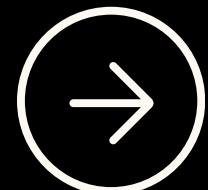
El análisis sintáctico, también llamado parsing, es una fase crucial del proceso de compilación en la que se verifica la estructura de un programa para asegurarse de que cumple con las reglas sintácticas definidas por el lenguaje de programación.

En esta etapa, el compilador toma como entrada una secuencia de tokens (producidos por la fase anterior, el análisis léxico) y construye una estructura, generalmente un árbol sintáctico, que refleja la jerarquía de las construcciones del lenguaje.





SINTAXIS



Qué es ?

Son un conjunto de reglas que definen cómo deben estructurarse las expresiones, sentencias y programas en un lenguaje determinado.

Para qué sirve ?

Estas reglas especifican cómo se pueden combinar símbolos, palabras clave y operadores del lenguaje para formar construcciones válidas, como declaraciones de variables, expresiones condicionales o bucles.

Ejemplo

```
int x = 5      ! Error de sintaxis  
';' expected
```

GRAMÁTICA



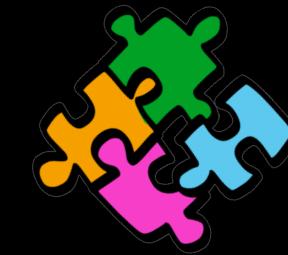
Qué es ?

Es un conjunto formal de reglas que define la estructura sintáctica correcta de un lenguaje de programación.



Cómo funciona ?

Durante la fase de análisis sintáctico (parsing) de un compilador, se utiliza una gramática para verificar que el programa fuente cumple con la estructura sintáctica correcta.



Ejemplo

Gramática para (1) + 1:

Expresión -> Expresión + Término

Expresión -> Término

Término -> Número

GRAMÁTICAS LIBRES DEL CONTEXTO (TIPO 2)

TERMINALES (T)

Conjunto de simbolos que conforman el alfabeto de nuestro lenguaje, conocidos tambien como Tokens.

PRODUCCIONES

En el lado **izquierdo** se tiene un **NT** llamado **encabezado**, en la parte **derecha** una secuencia de **T** y **NT** llamado **cuerpo**.

NO TERMINALES (NT)

Conjunto de variables sintácticas que representan un conjunto de cadenas o terminales.

PRODUCCIÓN INICIAL

Designación del símbolo o producción inicial.

GRAMÁTICA NO RECURSIVA

TERMINALES = NUM, +, -, *, /
NO TERMINALES = CALC, E

$$\begin{array}{l} \text{CALC} \rightarrow E + E \\ | \\ | \quad E - E \\ | \\ | \quad E * E \\ | \\ | \quad E / E \\ E \rightarrow \text{NUM} \end{array}$$

1 + 2, 3 - 1, 4 * 4, 3 / 5

GRAMÁTICA RECURSIVA

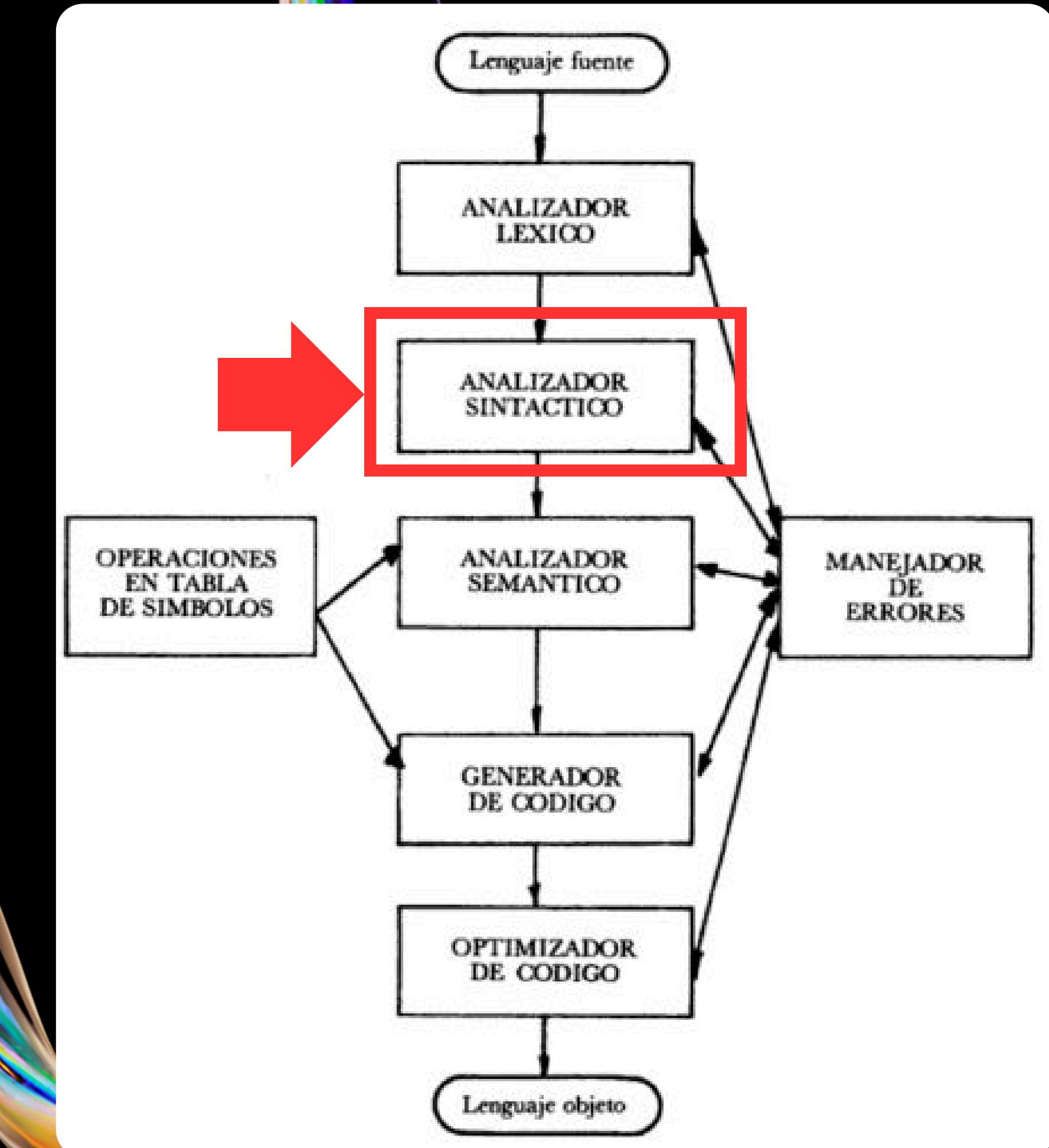
TERMINALES = NUM, +, -, *, /
NO TERMINALES = E

$$\begin{array}{l} E \rightarrow E + E \\ | \\ | \quad E - E \\ | \\ | \quad E * E \\ | \\ | \quad E / E \\ | \\ E \end{array}$$

E → NUM

1 + 2, 3 - 1, 4 * 4, 3 / 5, 3 + 5 * 2 / 3

FASES DEL COMPILADOR



ANALIZADOR O PARSER

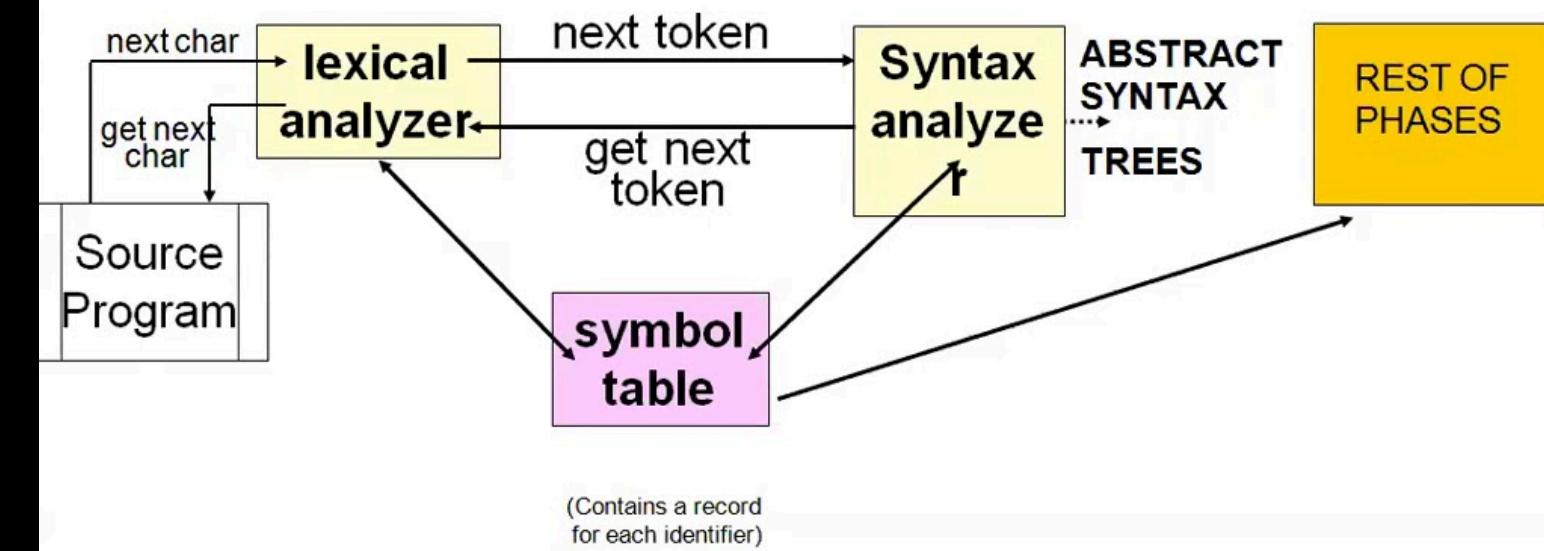
PARSER

Obtiene una cadena de tokens del analizador léxico y verifica que la cadena pueda generarse mediante la gramática definida para el lenguaje fuente.

Construye un árbol de análisis sintáctico y lo pasa por el resto del compilador para que se siga procesando.

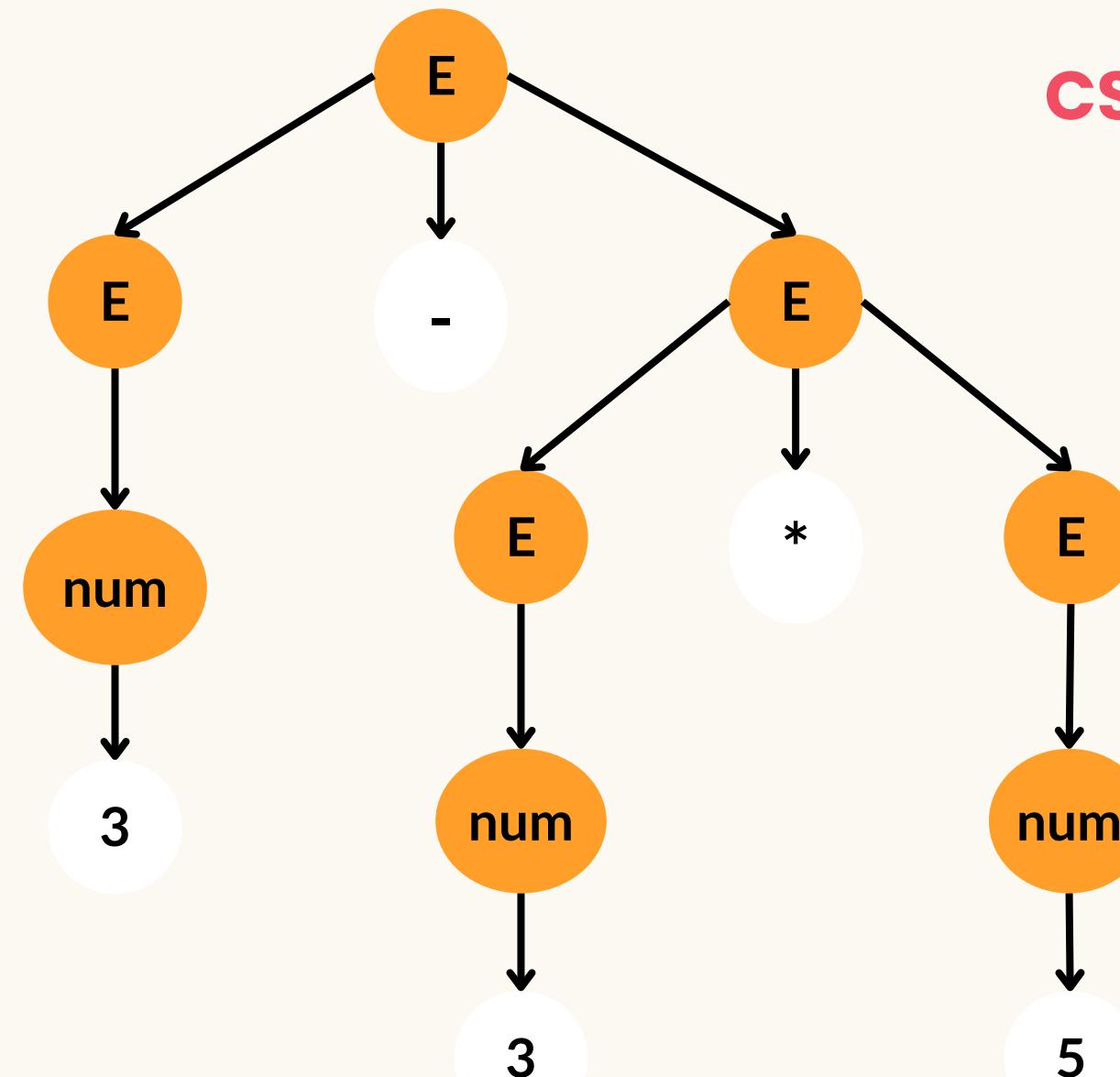
También reporta y se recupera de errores

Role of Parser



token: smallest meaningful sequence of characters of interest in source program

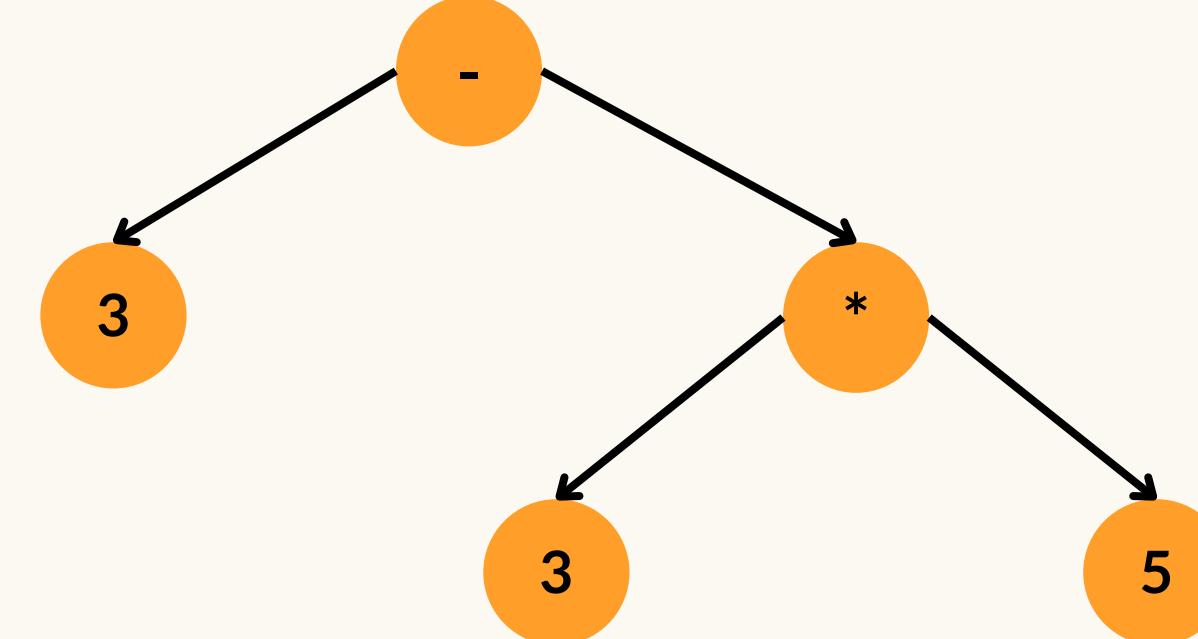
ÁRBOL DE ANÁLISIS SINTÁCTICO



CST

$$3 - 3 * 5 = ?$$

ÁRBOL DE SINTAXIS ABSTRACTA



AST

$$3 - 3 * 5 = ?$$

TRATAMIENTO DE ERRORES



Modo Panico

Al encontrar un error sintáctico se descartan los token siguientes (uno a la vez) hasta encontrar un token de sincronización, normalmente el ';' o un '}'.
** Garantiza no entrar en un ciclo infinito.

Nivel Frase

Se puede realizar una corrección local sobre la entrada restante sustituyendo un prefijo de dicha entrada por otra que le permita continuar.

Produccion de Error

Se trata de anticipar los errores comunes agregando producciones con las construcciones "erróneas" en la gramática del lenguaje.

Correccion Global

Dada una entrada x incorrecta, los algoritmos buscan encontrar un árbol de análisis sintáctico (CST) para una cadena y relacionada, de tal forma que con la gramática G se pueda transformar de x a y .



FIUSAC

EJEMPLO

CONSTRUCCIÓN DE UN PARSER

JAVA

