UNIVERSIDAD AUTÓNOMA DE CHIAPAS
FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN. CAMPUS I.
LICENCIATURA EN INGENIERÍA EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE.

COMPILADORES

ACT. 1.1 INVESTIGAR ANALIZADOR LÉXICO Y LENGUAJES REGULARES.

Ana Gabriela Casanova Hernández

Docente:
DR. LUIS GUTIÉRREZ ALFARO.

Tuxtla Gutiérrez, Chiapas. Domingo, 18 de agosto de 2024

Funciones del analizador léxico

El analizador léxico es la primera fase de un compilador. Su principal función consiste en leer los caracteres de entrada y elaborar como salida una secuencia de componentes léxicos que utiliza el analizador sintáctico para hacer el análisis.

Esta interacción suele aplicarse convirtiendo al analizador léxico en una subrutina o corrutina del analizador sintáctico. Recibida la orden "Dame el siguiente componente léxico" del analizador sintáctico, el léxico lee los caracteres de entrada hasta que pueda identificar el siguiente componente léxico, el cual devuelve al sintáctico según el formato convenido.

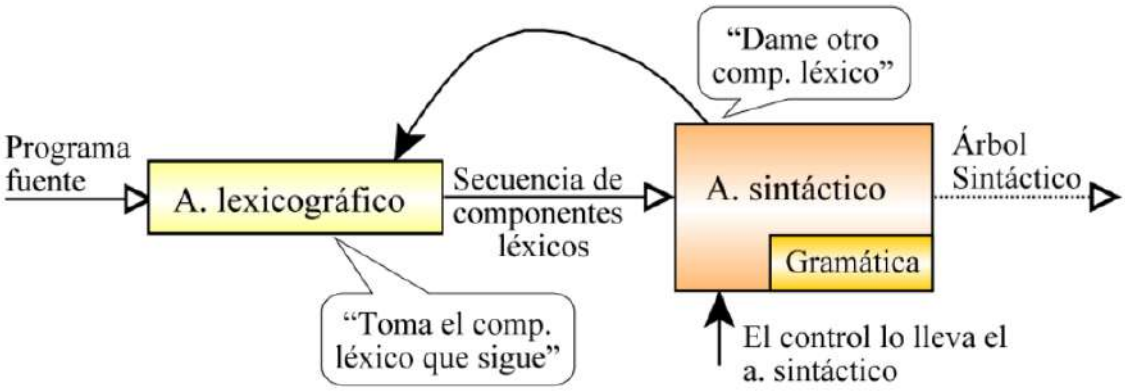

Figura 2.2. La fase de análisis léxico se halla bajo el control del análisis sintáctico. Normalmente se implementa como una función de éste.

Componentes léxicos, patrones y lexema

En la fase de análisis, los términos componentes léxicos (token), patrón y lexema se emplean con significados específicos. Un analizador léxico, inicialmente lee los lexemas y le asigna un significado propio.

• ***componente léxico*** es la secuencia lógica y coherente de caracteres relativo a una categoría: identificador, palabra reservada, literales (cadena/numérica), operador o carácter de puntuación, además de que un componente léxico puede tener uno o varios lexemas.

• ***patrón*** es una regla que genera la secuencia de caracteres que puede representar a un determinado componente léxico (expresión regular).

• ***lexema*** es una cadena de caracteres que concuerda con un patrón que describe un componente léxico (valor de cadena).

Ejemplo de una cadena de código: const pi = 3.1416;

LEXEMAS	COMPONENTES LEXICO	PATRON
Const	Const	const
=	Relación	< o <= o = o <> o > o >=
Pi	Identificador	Letra seguida de letras o números
3.1416	Numero	Cualquier literal numérica
“hola mundo”	literal	Caracteres entre comillas

El analizador léxico recoge información sobre los componentes léxicos en sus atributos asociados. Los tokens influyen en las decisiones del análisis sintáctico, y los atributos, en la traducción de los tokens.

En la práctica los componentes léxicos suelen tener solo un atributo. Para efectos de diagnóstico, puede considerarse tanto el lexema para un identificador como el número de línea en el que se encontró por primera vez. Esta información puede ser almacenada en la tabla de símbolos para el identificador (estructura de datos).

Para la cadena E=M*C**2 de ejemplo, los componentes léxicos y los valores de atributo asociado son:

<identificador, atributo para el símbolo E>
<op_asignacion>
<identificador, atributo para el símbolo M>
<op_multiplica>
<identificador, apuntador al símbolo C>
<op_exponente>
<numero, atributo valor 2>

Tome en cuenta que ciertas parejas no necesitan un valor de atributo. Los atributos relacionados con ese token deberán ser conservados y transferidos a alguna estructura de datos para que sean empleados en las siguientes etapas del análisis.

Tema **Lenguajes Regulares.**

Explicar el Lema de Bombeo para lenguajes regulares con un ejemplo.

El lema del bombeo Si L es un lenguaje regular, entonces existe una constante n, que depende

de L tal que para cada cadena w en L si |w| ≥ n, entonces podemos separar w en tres cadenas, w = xyz,

tal que:

1. y≠€
2. |xy| ≤ n

3. Para todas las k ≥ 0, la cadena xykz est´a tambi´en en L.

Lo anterior quiere decir que siempre podemos encontrar una cadena y no muy lejos del inicio de w que puede ser “bombeada” (o sea repetir y k veces o borrarla en el caso de que k = 0, y la cadena resultante sigue estando en el lenguaje L.

Explicar las propiedades de cerradura de lenguajes regulares con un ejemplo.

En esta sección se mostraran algunos teoremas de la forma “si ciertos lenguajes son regulares, y un lenguaje L está formado a partir de ellos mediante ciertas operaciones, entonces L es regular también”.

• ***Unión***. Si L y M son lenguajes regulares, entonces $L \cup M$ tambien lo es.

Si L1 y L2 son lenguajes regulares, L1 ∪ L2 también lo es.

Ejemplo: $L_1 = \{a\}, L_2 = \{b\} \rightarrow L_1 \cup L_2 = \{a, b\}$

• ***Complemento***. Si L es un lenguaje regular sobre el alfabeto Σ, entonces $\bar{L} = \Sigma * - L$ es tambien un lenguaje regular.

• ***Intersección***. Si L y M son lenguajes regulares, entonces $L \cap M$ es regular también.

• ***Diferencia***. Si L y M son lenguajes regulares, entonces $L - M$ es regular.

• ***Inversión***. Una cadena L se invierte al escribirla al revés para dar lugar a L^R. Si L es regular, L^R tambien lo es.

Explicar las propiedades de decisión de lenguajes regulares con un ejemplo.

En esta sección se considerar a la manera de responder preguntas sobre lenguajes regulares, pero antes hay que notar que un lenguaje puede ser infinito, por lo que responder la pregunta no debe requerir analizar un conjunto infinito de caracteres. En su lugar se plantea la pregunta usando una representación del lenguaje: DFA, NFA, -NFA o una expresión regular. Antes de comenzar a estudiar las técnicas y algoritmos para responder preguntas sobre lenguajes regulares con un repaso de las formas en las que se pueden hacer conversiones entre representaciones para un mismo lenguaje regular, poniendo especial atención en la complejidad temporal de los algoritmos que realizan las conversiones.

Explicar el proceso de determinación de equivalencias entre estados y lenguajes regulares con un ejemplo

Supongamos dos lenguajes:

- $L_1 = \{a^n b \mid n \geq 0\}$
- $L_2 = \{a^n b \mid n \geq 0\}$

Ambos lenguajes son iguales. Para verificar su equivalencia, construimos los autómatas para L1 y L2 y vemos que son idénticos

Explicar el proceso de minimización de DFA

El proceso de minimización de un DFA consiste en reducir el número de estados sin cambiar el lenguaje que acepta. Los pasos son:

Pasos de minimización:

1. Eliminar estados inalcanzables: Remueve los estados que no se pueden alcanzar desde el estado inicial.
2. Dividir en estados finales y no finales: Separar estados finales de no finales, ya que no pueden ser equivalentes.
3. Comparar estados equivalentes: Comparar pares de estados y agrupar aquellos que tienen las mismas transiciones para todos los símbolos del alfabeto.
4. Construir el DFA minimizado: Agrupa los estados equivalentes en un solo estado y ajusta las transiciones.

BIBLIOGRAFIA

- Compiladores, E., & Serna-Pérez. (n.d.). *UAA -Sistemas Electrónicos*. Retrieved August 19, 2024, from https://hopelchen.tecnm.mx/principal/sylabus/fpdb/recursos/r135468.PDF
- *Ciencias computacionales Propedeutico: Autómatas Propiedades de los Lenguajes Regulares*. (n.d.). https://posgrados.inaoep.mx/archivos/PosCsComputacionales/Curso_Propedeutico/Automatas/04_Automatas_PropiedadesLenguajesRegulares/CAPTUL1.PDF
- Compiladores.pdf. (2019). *Compiladores.pdf*. Google Docs. https://drive.google.com/file/d/1pI12XXGsI6z028sslyB-D9pR_CJEXIpV/view
- *Java a Tope: Traductores Y Compiladores Con Lex/yacc, Jflex/cup Y Javacc*. (2024). Google Books. https://books.google.es/books?id=F3lWLs1iTAMC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false