Universidad Autónoma de Chiapas UNACH

Facultad:

Contaduría y Administración, Campus I

Licenciatura:

Ingeniería en Desarrollo y Tecnologías de Software

Unidad de Aprendizaje:

Taller de desarrollo 4

Docente:

D.s.c. Luis Gutiérrez Alfaro

Alumno:

Lozano Monjarás David José

Grado y Grupo:

6° “M”

Matricula:

A210116

Actividad:

Lenguajes Regulares

Tuxtla Gutiérrez, Chiapas

Fecha: 23 – Enero – 2024

Contenido

1. Introducción

Un analizador léxico desempeña un papel fundamental en el proceso de compilación de un programa informático. Su función principal es escanear el código fuente de un programa y dividirlo en unidades más pequeñas llamadas "lexemas" o "tokens". Estos lexemas son representaciones simbólicas de elementos léxicos como palabras clave, identificadores, operadores y constantes. Además se ocupa de ciertas labores de “limpieza”. Entre ellas esta eliminar los blancos o los comentarios. También se ocupa de los problemas que pueden surgir por los distintos juegos de caracteres o si el lenguaje no distingue mayúsculas y minúsculas.

2. Funciones del analizador léxico

- Gestionar el fichero que contiene el código fuente, entendiéndose por ello el abrirlo, leerlo y cerrarlo
- Eliminar comentarios, tabuladores, espacios en blanco, saltos de línea.
- Relacionar los errores con las líneas del programa
- Expansión de macros.
- Inclusión de ficheros.
- Reconocimiento de las directivas de compilación.

- Introducir identificadores en la tabla de símbolos (esta función es opcional, pudiendo realizarse también por parte del analizador sintáctico).

3. Componentes (lexico,patrones,lexema)

Componente léxico = token + atributos.

Token: elemento sintáctico que describe el tipo de componente léxico (símbolos terminales de la gramática).

Patrón (asociado a un token): conjunto de cadenas para las que se obtiene el mismo token.

Lexema: secuencia de caracteres en el programa fuente con la que concuerda el patrón asociado a un token.

Componente léxico	Lexemas	Patrón
If	If	"if"
Mas	+	"+"
Identificador	velocidad, pi	Letra seguida de letras y dígitos
Número	10, 2000, 34	Secuencia de dígitos

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

El lema de bombeo se usa para demostrar que un Lenguaje No es Regular, es decir, que no puede ser aceptado, ese lenguaje, por un autómata finito determinístico**. Ejemplo:** Supongamos que tenemos el lenguaje L={0n1n|n≥0}, que consiste en todas las cadenas de ceros seguidas de la misma cantidad de unos. Queremos demostrar que este lenguaje no es regular.

- Supongamos que L es regular y tiene una constante de bombeo p.
- Tomamos la cadena s=0p1p, que claramente pertenece a L ya que tiene la forma 0n1n con =n=p.
- Según el lema del bombeo, podemos dividir s en xyz cumpliendo las condiciones del lema. Dado que ||≤|xy|≤p, las primeras p posiciones de s deben estar en y.
- Ahora, bombeamos y al menos una vez (tomamos i=2, por ejemplo). Esto significa que duplicamos la cantidad de ceros en xy2z.

- Sin embargo, en xy2z, la cantidad de unos no puede ser igual a la cantidad de ceros, ya que la parte y contiene solo ceros y los ceros se duplican. Por lo tanto, xy2z no puede pertenecer a L.
- Esto contradice la propiedad del lema del bombeo, lo que implica que L no es regular.

Con esto podemos ver cómo se puede utilizar el lema del bombeo para demostrar que un lenguaje no es regular al encontrar una cadena que no cumple con las condiciones del lema.

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

Algunas operaciones sobre lenguajes regulares garantizan producir lenguajes regulares:

Unión: L ∪ M lenguajes con cadenas en L, M o en ambos.

Intersección: L ∩ M lenguajes con cadenas en ambos.

Complemento: L cadenas que no están en L.

Diferencia: L\M o L − M.

Inversión: LR = {wR : w ∈ L}

Cerradura: L∗

Concatenación: L.M

Homomorfismo (sustitución): h(L) = {h(w) ∈ L}h es un

Homomorfismo.

Homomorfismo inverso (sustitución inversa):

h−1 (L) = {w ∈ Σ : h(w) ∈ L, h : Σ →} es un homomorfismo.

Unión: la unión de lenguajes regulares es regular. Sea L = L(E) y M = L(F). Entonces L(E + F) = L ∪ M , por la definición de “+” en RE.

Complemento: Si L es un lenguaje regular sobre Σ, entonces también lo es L = Σ∗ L. Todos los estados son de aceptación excepto los F.

Ejemplo: Sea L definido por el siguiente DFA (el lenguaje de cadenas que terminan en 01): Las cadenas de un número diferente de 0’s y 1’s es difícil probarlo con el pumping lemma. Sin embargo, ya probamos que L = 0n1 n no es regular. M = 0n1 m, n 6= m es L. Como L no es regular su complemento tampoco.

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

- Emptiness (Vacuidad):

Problema de decisión: Dado un lenguaje regular L, ¿es vacío?

Propiedad: Los lenguajes regulares son cerrados bajo la prueba de vacuidad. Es decir, hay un algoritmo que puede determinar si un lenguaje regular es vacío.

Ejemplo: Sea = L={0,01,001}. La prueba de vacuidad verificaría si hay alguna cadena en L. En este caso, L no es vacío.

- Finiteness (Finitud):

Problema de decisión: Dado un lenguaje regular L, ¿es finito?

Propiedad: Los lenguajes regulares son cerrados bajo la prueba de finitud. Existe un algoritmo que puede determinar si un lenguaje regular es finito.

Ejemplo: Sea L={0n|n≥0}. Este lenguaje representa todas las cadenas de ceros de longitud n. La prueba de finitud confirmaría que L es infinito.

- Igualdad de Lenguajes Regulares:

Problema de decisión: Dados dos lenguajes regulares L1 y L2, ¿son iguales?

Propiedad: Los lenguajes regulares son cerrados bajo la prueba de igualdad. Hay un algoritmo que puede verificar si dos lenguajes regulares son idénticos.

Ejemplo: Sean L1={0,10,110} y L2={10,0,110}. La prueba de igualdad verificaría que L1 y L2 son los mismos conjuntos, independientemente del orden.

- Inclusión de Lenguajes Regulares:

Problema de decisión: Dados dos lenguajes regulares L1 y L2, ¿es L1 un subconjunto de L2?

Propiedad: Los lenguajes regulares son cerrados bajo la prueba de inclusión. Existe un algoritmo que puede verificar si un lenguaje regular es un subconjunto de otro.

Ejemplo: Sea L1={0,01,001} y L2={0,01,001,10}. La prueba de inclusión confirmaría que L1 es un subconjunto de L2.

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

Se refiere al proceso de construcción de autómatas finitos deterministas (AFD) equivalentes a expresiones regulares o autómatas finitos no deterministas (AFN). Este proceso se conoce como determinización y es fundamental en la teoría de la computabilidad y en la implementación de compiladores. **Un ejemplo seria:**

- Consideraremos la expresión regular (a|b)*abb: Expresión Regular a AFN:
- Se construye un Autómata Finito No Determinista (AFN) equivalente para la expresión regular (a|b)*abb.

- El AFN podría tener transiciones que representan las opciones de a o b, y una secuencia específica para la cadena final abb.
- AFN a AFD (Determinización) - Algoritmo de Subconjuntos:
- Inicialmente, tomamos el conjunto de estados alcanzables por la cerradura-épsilon del estado inicial del AFN. En este caso, solo hay un estado inicial.
- Luego, para cada símbolo de entrada (a y b en este caso), calculamos el conjunto de estados alcanzables.
- Repetimos este proceso hasta que no se puedan agregar más conjuntos de estados.
- Después de aplicar el algoritmo de subconjuntos, obtenemos un AFD con estados más deterministas que representa el mismo lenguaje.
- Minimización del AFD (Opcional):
- En algunos casos, se puede aplicar un paso adicional de minimización para reducir el número de estados del AFD.
- Prueba de Equivalencia:
- Verificamos que el AFD generado sea equivalente al AFN original o a la expresión regular original.
- En este caso, ambos deberían aceptar o rechazar las mismas cadenas de entrada.

8. Explicar el proceso de minimización de DFA

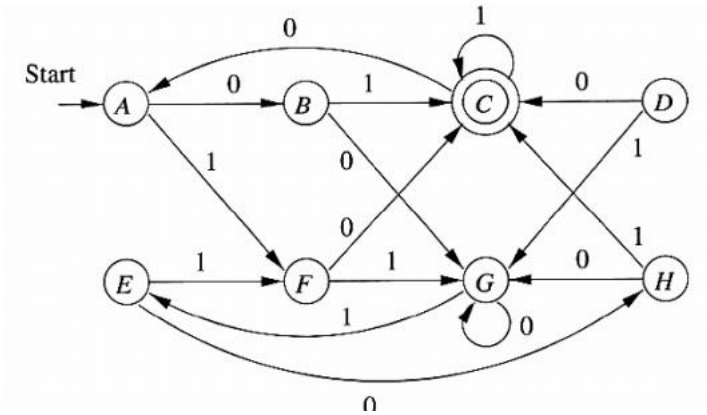
Una consecuencia de encontrar estados equivalentes, es que podemos reemplazar los estados equivalentes por un solo estado representado por su unión.

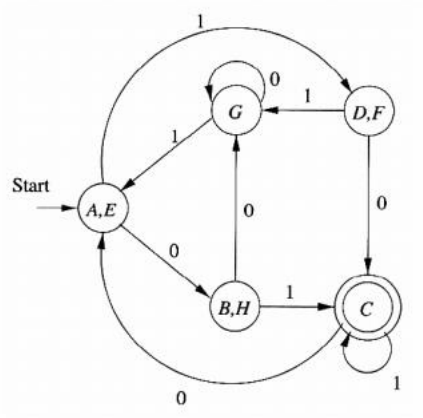
Por otro lado, la relación de equivalencia cumple con ser reflexiva, simétrica y transitiva.

La prueba de equivalencia de estados es transitiva, ósea que si encontramos que p y q son equivalentes y q y r son equivalentes, entonces p y r son equivalentes.

Ejemplo:

Minimizar el siguiente autómata usando los estados equivalentes.





Nota: no podemos aplicar el algoritmo de llenado de tabla a un NFA.

Suponga que se tiene la siguiente tabla de transción:

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
$*D$	D	A
E	D	F
F	G	E
G	F	G
H	G	D

- Hacer la tabla de estados distinguibles
- Construir un DFA de estados mínimos equivalente

9. Conclusión

Gracias a los conceptos y ejemplos vistos sabemos ahora que los analizadores léxicos desempeñan un papel crucial en el ámbito de la informática y la programación, especialmente en el desarrollo de compiladores y analizadores de lenguajes de programación. Son una parte esencial del proceso de compilación y análisis de código fuente, proporcionando beneficios significativos en términos de eficiencia, mantenibilidad y detección de errores en el desarrollo de software. Su importancia radica en su capacidad para transformar el código fuente en una forma más manejable y comprensible para el compilador o intérprete.

10. Bibliografías Formato APA

- *LEMA DE BOMBEO PARA LOS LENGUAJES REGULARES*. (s/f). Ugr.es. Recuperado el 24 de enero de 2024, de https://ccia.ugr.es/~rosa/tutormc/teoria/LEMA%20DE%20BOMBEO.htm
- de Ciencias Computacionales, P. (s/f). *Propiedades de los lenguajes regulares*. Inaoep.mx. Recuperado el 24 de enero de 2024, de https://posgrados.inaoep.mx/archivos/PosCsComputacionales/Curso_Propedeutico/Automatas/04_Automatas_PropiedadesLenguajesRegulares/PRESEN1.PDF
- Lexico, A. (s/f). *II26 Procesadores de lenguaje*. Uji.es. Recuperado el 24 de enero de 2024, de https://repositori.uji.es/xmlui/bitstream/handle/10234/5877/lexico.apun.pdf?sequence=1
- Perez, J. [@JorgePerez-pt3qg]. (2020, abril 23). *Teoría de la Computación - Clase 8: Equivalencia entre Expresiones Regulares y Autómatas Finitos*. Youtube. https://www.youtube.com/watch?v=FHveV-bs8s4