Universidad Autónoma de Chiapas
Facultad de contaduría y administración C-I

Carrera:
Lic. En ing en desarrollo y tecnologías de software.

Materia:
Compiladores.

Catedrático:
Mtro. Luis Gutiérrez Alfaro.

Nombre del alumno:
González Aguilar Eduardo - A211154

Semestre: 6. **Grupo:** M.

Nombre de la actividad:
investigación de componentes de expresiones regulares.

Fecha de entrega:
19/08/2023.

¿Qué es el analizador léxico?

Un **analizador léxico** o **analizador lexicográfico** (en inglés *scanner* o *tokenizer*) es la primera fase de un compilador, consistente en un programa que recibe como entrada el código fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de *tokens* (componentes léxicos) o símbolos.

Funciones del analizador léxico.

- Lee los caracteres de entrada y elaborar como salida una secuencia de componentes léxicos que utiliza el analizador sintáctico para hacer el análisis.

- Suele aplicarse convirtiendo al analizador léxico en una subrutina o co-rutina del analizador sintáctico.

- El analizador léxico lee los caracteres de entrada hasta que pueda identificar el siguiente componente léxico.

Funciones secundarias.

- Eliminar del programa fuente comentarios y espacios en blanco en forma de caracteres de espacio en blanco, caracteres TAB y de línea nueva.

- Otra función es relacionar los mensajes de error del compilador con el programa fuente.

- Se encarga de hacer una copia del programa fuente en el que están marcados los mensajes de error. Si el lenguaje fuente es la base de algunas funciones de preprocesamiento de macros, entonces esas funciones del preprocesador también se pueden aplica

Lenguajes regulares.

¿Qué son los lenguajes regulares?

Un lenguaje regular es un lenguaje formal que puede ser definido por una expresión regular, generado por una gramática regular, y reconocido por un autómata finito. Los lenguajes regulares son los lenguajes formales más sencillos en la Jerarquía de Chomsky

Lema de bombeo para lenguajes regulares.

El lema del bombeo para lenguajes regulares describe una propiedad esencial de todo lenguaje regular. Informalmente, dice que cualquier palabra suficientemente larga en un lenguaje regular puede ser bombeada - eso es, repetir una sección en la mitad de la palabra un número arbitrario de veces - para producir una nueva palabra que también pertenece al mismo lenguaje[1].

Por ejemplo, si consideramos el lenguaje $L = \{a^n b^n \mid n \geq 0\}$, podemos demostrar que no es regular usando el lema del bombeo. Supongamos que L es regular. Entonces, por el lema del bombeo, existe un número entero p (la longitud de bombeo) tal que cualquier cadena w en L con $|w| \geq p$ puede ser escrita como $w = xyz$ con las siguientes propiedades:

1. $|xy| \leq p$
2. $|y| > 0$

3. Para todo $i \geq 0$, $xy^iz \in L$

Tomemos la cadena $w = a^p b^p \in L$. Por (1), podemos escribir $w = xyz$ donde $x = a^k$, $y = a^l$ y $z = a^{(p-k-l)} b^p$ para algunos enteros k y l con $k + l \leq p$. Por (2), sabemos que y contiene al menos un símbolo a. Tomemos $i = 0$. Entonces $xy^0z = xz = a^{(p-k-l)} b^p \notin L$, lo cual contradice (3). Por lo tanto, L no es regular.

Propiedades de cerradura de lenguajes regulares.

Cerradura bajo la unión:

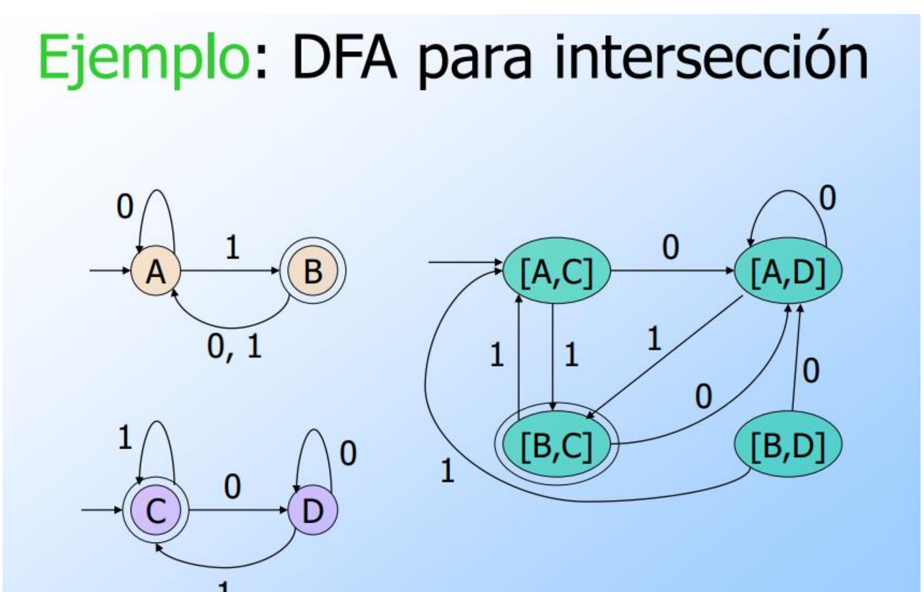
- Si L y M son lenguajes regulares, también lo es $L \cup M$.
- Prueba: Sean L y M lenguajes representados por las expresiones regulares R y S, respectivamente.
- Entonces, R+S es una expresión regular cuyo lenguaje es $L \cup M$

Cerradura de la concatenación y de Kleene:

- RS es una expresión regular cuyo lenguaje es LM.
- R* es una expresión regular cuyo lenguaje es L*

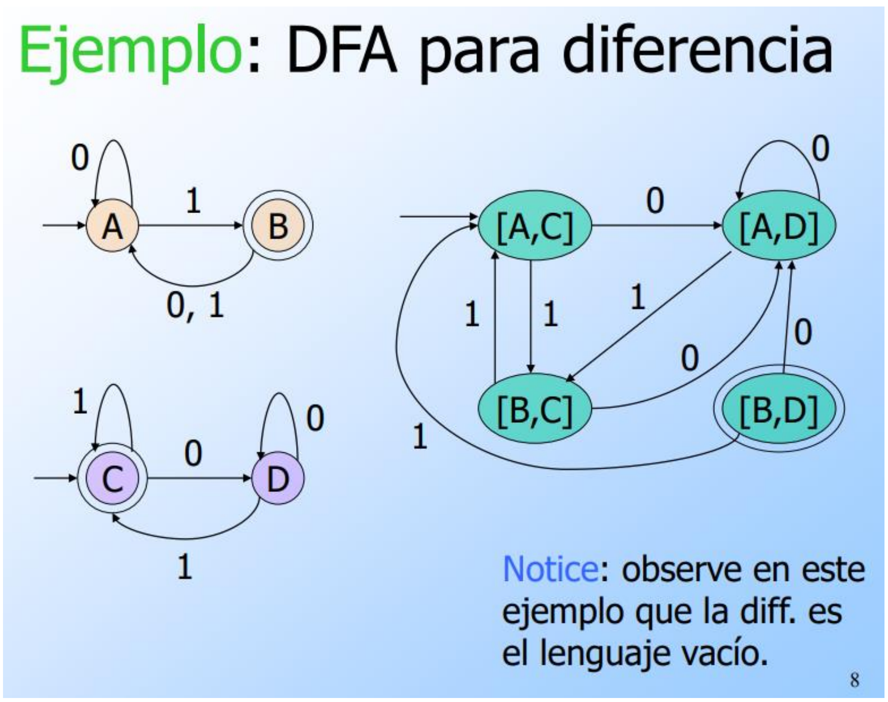
Cerradura de la intersección:

- Si L y M son lenguajes regulares, entonces también lo es $L \cap M$.
- Prueba: Sean A y B dos autómatas AFD cuyos lenguajes son L y M, resp.
- construimos $C = A \times B$, el autómata producto de A y B.
- Hacemos los estados finales de C, aquellos pares [q, r], donde q es estado final de A, y r es estado final de B.



Cerradura bajo la diferencia:

- Si L y M son lenguajes regulares, entonces también lo es L – M . (las cadenas en L, pero no en M).
- Prueba: Sean A y B autómatas AFD cuyos lenguajes son L y M, resp.
- Construimos C=A×B, el autómata producto.
- Hacemos los estados finales de C, aquellos pares [q, r], donde q es estado final de A, pero r no es estado final de B



Cerradura bajo complemento:

- El complemento de un lenguaje L (con respecto al alfabeto Σ, con Σ* conteniendo L) es Σ* – L.
- Como Σ* es ciertamente un lenguaje regular, el complemento de un lenguaje regular L es también un lenguaje regular

Cerradura bajo reserva:

- Dado un lenguaje L, LR es el conjunto de todas las cadenas cuya cadena reversa está en L.
- Ejemplo: L = {0, 01, 100}; L R = {0, 10, 001}.
- Sea E una expresión regular para L.
- Mostraremos cómo revertir E, y producir una expresión regular ER para LR.

Cerradura bajo homomorfismos:

- Si L es un lenguaje regular, y h es un homomorfismo sobre su alfabeto Σ, entonces h(L) = {h(w): w ϵ L} es también un lenguaje regular.
- Prueba: Sea E una expresión regular de L. Aplicar h a cada símbolo en E.
- El lenguaje de la regexp resultante es h(L).

Propiedades de cerradura de lenguajes regulares.

Los lenguajes regulares tienen varias propiedades de decisión. Algunas de ellas son:

- Pertenencia: ¿Pertenece L1 a L2?
- Intersección vacía: ¿L1 ∩ L2 es vacío?
- Lenguaje vacío: ¿Es L1 vacío?
- Pertenencia: Dado w que pertenece a Σ*, ¿está w en L1?

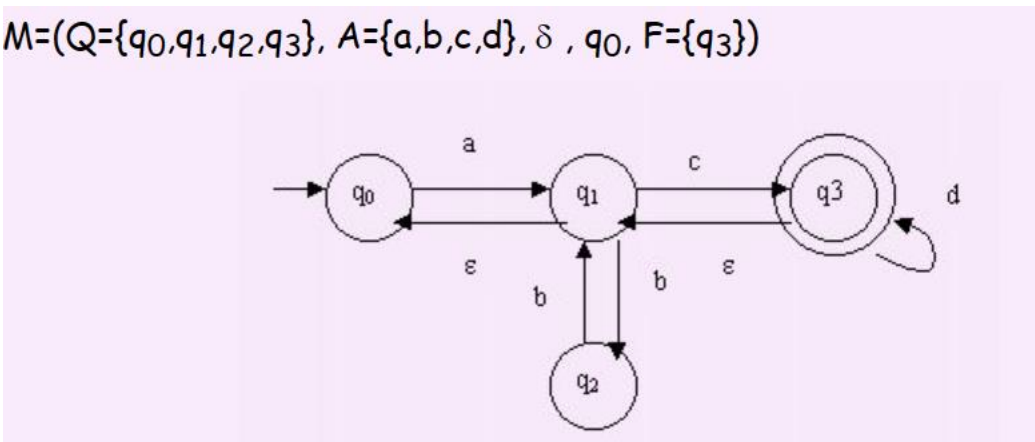
Un ejemplo de la propiedad de decisión de pertenencia sería el siguiente:

Si A y B son autómatas finitos deterministas con L(A) y L(B) los lenguajes que son aceptados por los autómatas respectivamente, entonces el problema de decidir si un elemento w pertenece a L(A) es decidible.

Proceso de determinación de equivalencias entre estados y lenguajes regulares.

El proceso de determinación de equivalencias entre estados y lenguajes regulares se puede realizar mediante la construcción de un autómata finito determinista (AFD) que reconozca el lenguaje regular en cuestión. Para ello, se puede seguir el siguiente proceso[1]:

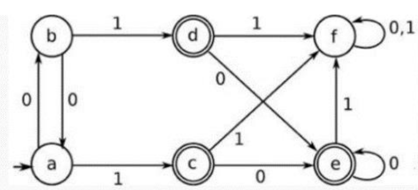
1. Se construye un autómata finito no determinista (AFND) que reconozca el lenguaje regular.
2. Se convierte el AFND en un AFD equivalente.
3. Se minimiza el AFD para obtener un autómata mínimo.



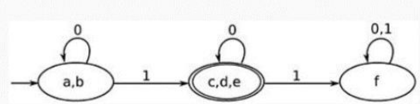
Proceso de la minimización de DFA

La minimización de un autómata finito determinista (DFA) se realiza normalmente en tres pasos[1]:

1. Eliminar estados muertos e inalcanzables.
2. Fusionar estados no distinguibles.
3. Opcionalmente, volver a crear un solo estado muerto (estado “receptor”) si se requiere que el DFA resultante esté completo.



Ejemplo DFA. Si está en el estado *c* , exhibe el mismo comportamiento para cada cadena de entrada que en el estado *d* , o en el estado *e* . Del mismo modo, los estados *una* y *b* son nondistinguishable. El DFA no tiene estados inalcanzables.



Bibliografía

Libro: Jeffrey D. (1998). *Compiladores: Principios, técnicas y herramientas*.

En *Fundamentos generales de programación*, de Luis Joyanes Aguilar, 33-37. México: Mc Graw-Hill, 2013.

Zwisnki Jamie.(2010). JavaScript: Expresiones regulares

Y. Bar-Hillel, M. Perles, E. Shamir, "On formal properties of simple phrase structure grammars", *Zeitschrift für Phonetik, Sprachweissenshaft und Kommunikationsforschung* **14** (1961) pp. 143-172.

Reyes Figueroa A.(17 agosto del 2022), Propiedades de cerradura de los lenguajes regulares, pp 1-10, pp14 Aula 08 (pfafner.github.io)