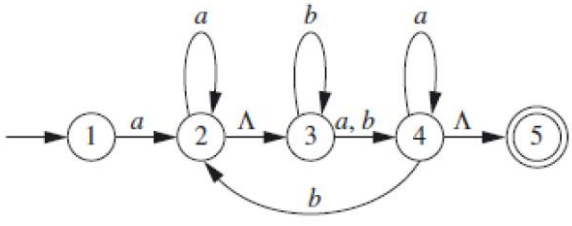
Universidad del Valle de Guatemala
Teoría de la Computación, CC2019
Docentes: Alan Reyes; Tomás Gálvez P.
Semestre 2, 2022

Tarea #1

1. ¿Qué son definiciones regulares?
2. ¿Qué es una especificación léxica?
3. ¿Cuál es la diferencia entre un componente (*token*) de una especificación léxica y un lexema?
4. Dadas dos expresiones regulares r_1 y r_2. ¿Cómo demostraría (o refutaría) que $L(r_1) = L(r_2)$? Demuestre que para una expresión regular cualquiera r, $L(r^*) = L(r^{**})$.
5. Escriba expresiones regulares que describan los siguientes lenguajes:
   a. *Strings* sobre el alfabeto $\{a, b, c\}$ donde la primera a siempre vaya inmediatamente antes de la primera b.
   b. *Strings* sobre el alfabeto $\{a, b, c\}$ con un número par de a's.
   c. Números binarios múltiplos de cuatro.
6. Según lo que hemos visto en clase, no hay expresiones regulares que describan los siguientes lenguajes:
   - *Strings* sobre el alfabeto $\{a, b\}$ donde hay más a's que b's.
   - Palíndromos sobre el alfabeto $\{a, b\}$.

   Proponga una explicación que responda por qué.

   Idea para desarrollar: cuando simulamos un autómata, ¿qué información nos dan los estados visitados en cada paso? ¿Qué información no nos dan?
7. ¿Cómo se define un AFN? ¿Cuál es la diferencia entre un AFN y un AFD? ¿Cuál es la diferencia entre AFN (o AFD) y diagrama de transición?
8. ¿Cuáles son los estados importantes en un AFN y cómo se relacionan con la construcción directa de un AFD a partir de una expresión regular?
9. Sea $w = xy$ donde x e y son cadenas sobre un conjunto de símbolos dados. Demuestre que $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$, donde $\hat{\delta}$ es la función de transición definida para un AFD.
10. Construya AFD's que acepten los siguientes lenguajes sobre el alfabeto {0,1}:
    a. Conjunto de cadenas que terminen en 00.
    b. Conjunto de cadenas con tres 0 consecutivos (no necesariamente al final).
    c. Conjunto de cadenas que empiecen con 01.
    d. Conjunto de cadenas que contengan a 001 como sub-cadena.
11. Convierta el siguiente AFN en expresión regular. Dibuje los pasos de su procedimiento de conversión. **Nota**: Λ es ε.



12. Reescriba la expresión regular del ejercicio anterior en notación *postfix* y dibuje el árbol sintáctico correspondiente.
13. Para las siguientes expresiones regulares construya un AFN por medio del algoritmo de Thompson y luego conviértalo a AFD por medio de construcción de subconjuntos.
    a. $0?\,(1|\varepsilon)?\,0^*$
    b. ab^*ab^*
    c. $((a|b)^*)^*\ \varepsilon\big((a|b)\big|\varepsilon\big)^*$
    d. $(a|b)^*\big(\big(a\big|(bb)\big)^*\varepsilon\big)$
14. Para cada expresión regular del ejercicio anterior:
    a. Construya un AFD directamente y minimice su cantidad de estados con cualquiera de los acercamientos vistos en clase (particiones o tablita). Comente sobre la comparación entre los AFD's construidos directamente y los minimizados.
    b. Minimice los estados de los AFD's obtenidos en el ejercicio anterior. Comente sobre la comparación entre los AFD's minimizados a partir de la conversión y a partir de la construcción directa.

Ponderación:

Ejercicios	Puntos
1, 4, 3, 7, 8	2
5, 10, 11, 12	5
4, 6, 9	10
13, 14	20

Fuentes:

- Aho, A. V., Lam, M. S., Ravi, S., & Ullman, J. D. (2007). *Compilers: Principles, Techniques and Tools.* Pearson.
- Appel, A. (2004). *Modern Compiler Implementation in (Java|C|ML).* Cambridge.
- Pojoy, B (2012). *Curso: Diseño de Lenguajes de Programación*. UVG.