Parcial Final, últimos 2 puntos

Lenguajes Formales y Compiladores (2025-2)

Group Number: C2566-ST0270-3952

Nombre Completo
Juan Manuel Young Hoyos

Tutor: Oscar Rodriguez Cifuentes

Medellín, 22 de noviembre de 2025

Índice

1 | Ejercicio 5 [20 pts]

Considere el siguiente lenguaje fuente L sobre el alfabeto $\Sigma = \{a, b\}$: todas las cadenas que contienen una o más repeticiones del patrón ab.

a. Diseñe una gramática libre de contexto G que genere L.

b. Extienda dicha gramática para construir una gramática de traducción G_τ que, por cada aparición del patrón ab, produzca como salida el símbolo X; es decir, si la entrada es $ababab$, entonces la salida es XXX.

c. Escriba una expresión de traducción regular equivalente a la gramática G_τ.

2 | Desarrollo Ejercicio 5

2.1 | a. Gramática libre de contexto G

Para generar el lenguaje L (cadenas con al menos una aparición de ab), utilizamos dos no terminales: S (estado inicial, buscando el primer ab) y F (estado de aceptación, el patrón ya fue encontrado), siguiendo los principios de diseño de gramáticas formales [2].

$$S \to aS \mid bS \mid abF$$
$$F \to aF \mid bF \mid abF \mid \epsilon$$

2.2 | b. Gramática de traducción G_τ

Basándonos en el esquema de traducción dirigido por sintaxis [1], extendemos la gramática anterior. Los caracteres que no forman parte del patrón ab se traducen a la cadena vacía (ϵ), y el patrón ab se traduce a X.

$$S \to \frac{a}{\epsilon}S \mid \frac{b}{\epsilon}S \mid \frac{ab}{X}F$$
$$F \to \frac{a}{\epsilon}F \mid \frac{b}{\epsilon}F \mid \frac{ab}{X}F \mid \epsilon$$

2.3 | c. Expresión de traducción regular

La expresión equivalente describe: cualquier prefijo irrelevante (se borra), seguido de un ab obligatorio (se vuelve X), seguido de cualquier sufijo (se procesa según corresponda) [2].

$$e_\tau = \left(\frac{a}{\epsilon} \cup \frac{b}{\epsilon}\right)^* \cdot \frac{ab}{X} \cdot \left(\frac{a}{\epsilon} \cup \frac{b}{\epsilon} \cup \frac{ab}{X}\right)^*$$

3 | Ejercicio 6 [20 pts]

Diseñe un transductor sequencial determinista M que lea cadenas sobre $\Sigma = \{0, 1\}$ y produzca como salida la complementaria de cada bit ($0 \to 1, 1 \to 0$); por ejemplo, si la entrada es 10110, entonces la salida debe ser 01001. Adicionalmente, dibuje el diagrama de estados del transductor y especifique formalmente las funciones de transición δ y de salida η.

4 | Desarrollo Ejercicio 6

4.1 | Análisis y Diseño

El objetivo es diseñar un transductor secuencial determinista que calcule el complemento a 1 de la entrada (intercambiar 0s y 1s). Prácticamente en resumen el problema solicita calcular el complemento a 1 bit a bit.

- **Análisis de Memoria:** Para decidir la salida de un bit actual, ¿necesitamos recordar el bit anterior? No. La transformación de $0 \to 1$ y $1 \to 0$ es independiente del contexto histórico de la cadena.
- **Conclusión:** Dado que no se requiere memoria (contexto), el autómata solo necesita un único estado (q_0) que actúe como bucle perpetuo procesando la entrada símbolo a símbolo. Esto garantiza que el transductor sea *determinista* y *secuencial* [2].

4.2 | Especificación Formal

El transductor se define como la tupla $M = (Q, \Sigma, \Delta, q_0, F, \delta, \eta, \varphi)$, siguiendo la definición formal de autómatas con salida [3], donde:

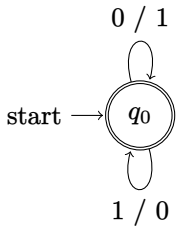
- $Q = \{q_0\}$ (Un único estado funcional).
- $\Sigma = \{0, 1\}$ (Entrada), $\Delta = \{0, 1\}$ (Salida).
- $F = \{q_0\}$ (El estado es final porque cualquier prefijo es válido).

Definición de Funciones

1. **Función de transición** ($\delta : Q \times \Sigma \to Q$): El autómata permanece en el mismo estado.
$$\delta(q_0, 0) = q_0, \quad \delta(q_0, 1) = q_0$$
Siempre regresamos al mismo estado para procesar el siguiente bit.
2. **Función de salida** ($\eta : Q \times \Sigma \to \Delta^*$): Emite el bit complementario.
$$\eta(q_0, 0) = 1, \quad \eta(q_0, 1) = 0$$
Aquí reside la lógica del complemento: si leemos 0 escribimos 1, y viceversa.
3. **Función final** ($\varphi : F \times \{\dashv\} \to \Delta^*$): No se emite nada al terminar la cadena.
$$\varphi(q_0, \dashv) = \epsilon$$
Al terminar la cadena, no hay bits pendientes ni operaciones de cierre (como acarreos), por lo que la salida final es vacía.

4.3 | Diagrama de Estados

El diagrama de transición de estados es entonces:



4.4 | Verificación (Traza)

Probamos el diseño con la cadena de ejemplo del enunciado: $w = 10110$.

Estado Actual	Entrada Leída	Salida Generada (η)	Siguiente Estado (δ)
q_0	1	0	q_0
q_0	0	1	q_0
q_0	1	0	q_0
q_0	1	0	q_0
q_0	0	1	q_0
	Salida Total:	**01001**	

Cuadro 4.1: Traza de ejecución para la entrada 10110

La salida obtenida 01001 coincide con el complemento a 1 solicitado.

5 | Referencias

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, & Tools*. Pearson/Addison Wesley, Boston, 2nd edition, 2007.

[2] Stefano Crespi Reghizzi, Luca Breveglieri, and Angelo Morzenti. *Formal Languages and Compilation*. Texts in Computer Science. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-04879-2.

[3] Dexter C. Kozen. *Automata and Computability*. Springer New York, New York, NY, 1997. doi:10.1007/978-1-4612-1844-9.