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]

1.1 | Enunciado

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.

1.2 | a. Gramática libre de contexto G

Una gramática libre de contexto $G = (\Sigma, N, P, S)$ que genera L es:

$$P : \begin{cases} S \to aS \mid bS \mid abF \\ F \to aF \mid bF \mid abF \mid \epsilon \end{cases}$$

Con símbolos no terminales $N = \{S, F\}$. **Justificación:** La producción S consume cualquier prefijo (ruido) hasta encontrar el primer patrón obligatorio ab, momento en el cual transiciona a F. El no terminal F permite generar repeticiones adicionales del patrón o terminar la cadena con ϵ, asegurando así la condición de una o más repeticiones [2].

1.3 | b. Gramática de traducción G_τ

Extendemos G para definir la gramática de traducción $G_\tau = (V, \Sigma, \Delta, P_\tau, S)$ definida a través de las producciones:

$$P_\tau : \begin{cases} S \to \frac{a}{\epsilon}S \mid \frac{b}{\epsilon}S \mid \frac{ab}{X}F \\ F \to \frac{a}{\epsilon}F \mid \frac{b}{\epsilon}F \mid \frac{ab}{X}F \mid \epsilon \end{cases}$$

Con símbolos no terminales $V = \{S, F\}$, símbolos terminales de destino $\Delta = \{X\}$ y el alfabeto terminal de pares $C = \{\frac{a}{\epsilon}, \frac{b}{\epsilon}, \frac{ab}{X}\} \subseteq \Sigma^* \times \Delta^*$.

Esquema de Traducción

A partir de G_τ obtenemos el siguiente esquema, separando la sintaxis de origen y la semántica de destino [1]:

Gramática de Origen	**Gramática de Destino**
$S \to aS \mid bS \mid abF$ $F \to aF \mid bF \mid abF \mid \epsilon$	$S \to \epsilon S \mid \epsilon S \mid XF$ $F \to \epsilon F \mid \epsilon F \mid XF \mid \epsilon$

Cuadro 1.1: Esquema de traducción asociado a G_τ

Verificación de la Traducción

Procesemos la traducción de la cadena de entrada $w = ababab$. Realizamos la derivación utilizando los pares de traducción:

$$\begin{aligned} S &\to \frac{ab}{X}F \\ &\to \frac{ab}{X}\frac{ab}{X}F \\ &\to \frac{ab}{X}\frac{ab}{X}\frac{ab}{X}F \\ &\to \frac{ab}{X}\frac{ab}{X}\frac{ab}{X}\epsilon \\ &= \frac{ababab}{XXX} \equiv z \end{aligned}$$

Sean h_Σ y h_Δ los homomorfismos alfabéticos que proyectan sobre los alfabetos de origen y destino respectivamente:

$$h_\Sigma(z) = ababab, \quad h_\Delta(z) = XXX$$

De esta manera, $(ababab, XXX) \in \rho_\tau$, confirmando que la traducción es correcta.

1.4 | c. Expresión de traducción regular (e_τ)

Primero, determinamos una expresión regular capaz de generar el lenguaje fuente $L(G)$:

$$(a|b)^*ab(a|b)^*$$

Sin embargo, para capturar **cada** ocurrencia de ab como pide la traducción (y no solo la primera), refinamos la expresión conceptualmente como: "Prefijo irrelevante, seguido de grupos de (patrón o ruido)". Una expresión regular de traducción e_τ equivalente es:

$$e_\tau = \underbrace{\left(\frac{a}{\epsilon} \mid \frac{b}{\epsilon}\right)^*}_{\text{Prefijo}} \cdot \frac{ab}{X} \cdot \underbrace{\left[\left(\frac{a}{\epsilon} \mid \frac{b}{\epsilon}\right)^* \frac{ab}{X}\right]^*}_{\text{Repeticiones}} \cdot \underbrace{\left(\frac{a}{\epsilon} \mid \frac{b}{\epsilon}\right)^*}_{\text{Sufijo}}$$

Esta estructura asegura que el primer ab se traduce, y cualquier ab subsecuente (mezclado con ruido) también se traduce a X.

2 | Ejercicio 6 [20 pts]

2.1 | Enunciado

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.

2.2 | 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].

2.3 | Especificación Formal

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

- $Q = \{q_0\}$ (conjunto de estados)
- $\Sigma = \{0, 1\}$ (alfabeto de origen)
- $\Delta = \{0, 1\}$ (alfabeto de destino)
- $F = \{q_0\}$ (conjunto de estados de aceptación)

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.

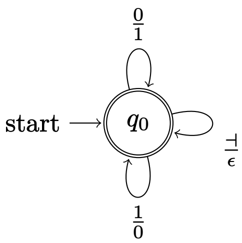
3. **Función final** ($\varphi : F \times \{\dashv\} \to \Delta^*$): No se emite nada al terminar la cadena.

$$\varphi(q_0, \dashv) = \epsilon$$

   Al terminar la cadena, no hay bits pendientes ni operaciones de cierre (como acarreos), por lo que la salida final es vacía.

2.4 | Diagrama de Estados

El diagrama de transición de estados es entonces:



2.5 | Verificación (Traza de ejecución)

Para verificar la corrección, procesamos la cadena de ejemplo $w = 10110$:

Estado	**Entrada**	**Salida (η)**	**Descripción**
q_0	1	0	Lectura de 1, complemento 0
q_0	0	1	Lectura de 0, complemento 1
q_0	1	0	Lectura de 1, complemento 0
q_0	1	0	Lectura de 1, complemento 0
q_0	0	1	Lectura de 0, complemento 1
q_0	$\dashv$	ϵ	Fin de cadena (φ)
Resultado Final:		**01001**	

Cuadro 2.1: Traza paso a paso para la entrada 10110

3 | 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.