



# 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, 19 de noviembre de 2025

# Índice

<b>1</b>	<b>Ejercicio 5 [20 pts]</b>	<b>2</b>
<b>2</b>	<b>Desarrollo Ejercicio 5</b>	<b>2</b>
2.1	a. Gramática libre de contexto $G$ . . . . .	2
2.2	b. Gramática de traducción $G_\tau$ . . . . .	2
2.3	c. Expresión de traducción regular . . . . .	2
<b>3</b>	<b>Ejercicio 6 [20 pts]</b>	<b>2</b>
<b>4</b>	<b>Desarrollo Ejercicio 6</b>	<b>2</b>
4.1	Definición Formal . . . . .	3
4.2	Funciones del Transductor . . . . .	3
4.3	Diagrama de Estados . . . . .	3
<b>5</b>	<b>Referencias</b>	<b>4</b>

## 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  $ab$ .

- Diseñe una gramática libre de contexto  $G$  que genere  $L$ .
- Extienda dicha gramática para construir una gramática de traducción  $G_\tau$  que, por cada aparición del patrón  $ab$ , produzca como salida el símbolo  $X$ ; es decir, si la entrada es  $ababab$ , entonces la salida es  $XXX$ .
- Escriba una expresión de traducción regular equivalente a la gramática  $G_\tau$ .

## 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].

$$\begin{aligned} S &\rightarrow aS \mid bS \mid abF \\ F &\rightarrow aF \mid bF \mid abF \mid \epsilon \end{aligned}$$

### 2.2 | b. Gramática de traducción $G_\tau$

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 ( $\epsilon$ ), y el patrón  $ab$  se traduce a  $X$ .

$$\begin{aligned} S &\rightarrow \frac{a}{\epsilon}S \mid \frac{b}{\epsilon}S \mid \frac{ab}{X}F \\ F &\rightarrow \frac{a}{\epsilon}F \mid \frac{b}{\epsilon}F \mid \frac{ab}{X}F \mid \epsilon \end{aligned}$$

### 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 secuencial determinista  $M$  que lea cadenas sobre  $\Sigma = \{0, 1\}$  y produzca como salida la complementaria de cada bit ( $0 \rightarrow 1, 1 \rightarrow 0$ ); 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  $\delta$  y de salida  $\eta$ .

## 4 | Desarrollo Ejercicio 6

El objetivo es diseñar un transductor secuencial determinista que calcule el complemento a 1 de la entrada (intercambiar 0s y 1s).

#### 4.1 | Definición 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:

- $Q = \{q_0\}$
- $\Sigma = \{0, 1\}, \Delta = \{0, 1\}$
- $F = \{q_0\}$

#### 4.2 | Funciones del Transductor

1. **Función de transición ( $\delta$ ):** El autómata permanece en el mismo estado.

$$\delta(q_0, 0) = q_0, \quad \delta(q_0, 1) = q_0$$

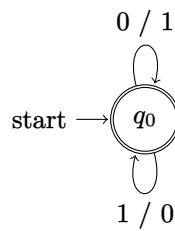
2. **Función de salida ( $\eta$ ):** Emite el bit complementario.

$$\eta(q_0, 0) = 1, \quad \eta(q_0, 1) = 0$$

3. **Función final ( $\varphi$ ):** No se emite nada al terminar la cadena.

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

#### 4.3 | Diagrama de Estados



## 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](https://doi.org/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](https://doi.org/10.1007/978-1-4612-1844-9).