

Teoría de Autómatas y Lenguajes Formales

Unidad 1 — Sesión 2 (09-ene-2026): AFD (diseño y verificación)

Docente: Helder Octavio Fernández Guzmán

- Un **lenguaje** es un conjunto de cadenas: $L \subseteq \Sigma^*$.
- Hoy veremos una herramienta para decidir, de forma mecánica: $w \in L$ o $w \notin L$?
- Esa herramienta será un **Autómata Finito Determinista (AFD)**.

Regla mental del día: leer símbolo por símbolo y hacer un **recorrido de estados** (paso a paso).

Al finalizar la sesión podrás:

- Explicar qué es un **AFD** y para qué sirve.
- Diseñar AFD para lenguajes regulares simples sobre $\Sigma = \{a, b\}$.
- Verificar un AFD con **simulación paso a paso** y casos de prueba.
- Usar **Automata Tutor** para simular y depurar (hoy guiado).

¿Qué es un AFD? (idea)

Periodo 2/6

Un AFD es una **máquina de estados** con memoria finita que:

- 1 empieza en un estado inicial,
- 2 lee la cadena símbolo por símbolo,
- 3 cambia de estado según transiciones,
- 4 al final **acepta** si termina en un estado de aceptación.

Determinista: para cada estado y cada símbolo, hay **una única** transición posible.

Definición mínima (formal, sin exceso)

Periodo 2/6

Un AFD se define como:

$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q : conjunto finito de estados
- Σ : alfabeto
- $\delta : Q \times \Sigma \rightarrow Q$ (función de transición)
- $q_0 \in Q$: estado inicial
- $F \subseteq Q$: estados de aceptación

Pertenencia: M acepta w si, al procesar w desde q_0 , termina en un estado de F .

Cómo verificar (paso a paso)

Periodo 2/6

Para verificar un AFD con una cadena $w = s_1 s_2 \cdots s_n$:

- 1 Inicia en q_0 .
- 2 Para cada símbolo s_i , actualiza: $q \leftarrow \delta(q, s_i)$.
- 3 Al final, si $q \in F$ entonces **acepta**; si no, **rechaza**.

En aula: lo haremos como **recorrido de estados**.

En Automata Tutor: lo haremos como **simulación paso a paso**.

Lenguaje 1 (ancla): termina en ab

Periodo 3/6

Definimos:

$$L_1 = \{ w \in \{a, b\}^* : w \text{ termina en } ab \}$$

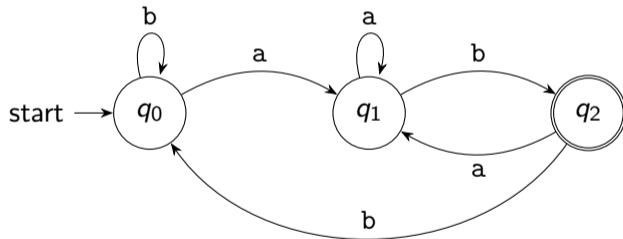
Ejemplos:

- **Acepta:** ab, aab, bbab
- **Rechaza:** aba, baba, bbb

Idea de diseño: ¿qué mínimo debo **recordar** del sufijo para decidir?

Diseño del AFD para L_1 (diagrama de estados)

Periodo 3/6



Significado de estados:

- q_0 : no termina en a
- q_1 : el último símbolo fue a
- q_2 : termina en ab (**acepta**)

Verificación paso a paso (recorrido de estados)

Periodo 3/6

Ejemplo con $w = aab$:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2$$

Conclusión: acepta, porque termina en $q_2 \in F$.

Ejemplo con $w = aba$:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_1$$

Conclusión: rechaza (termina en $q_1 \notin F$).

Lenguaje 2: contiene la subcadena aa

Periodo 4/6

Definimos:

$$L_2 = \{ w \in \{a, b\}^* : w \text{ contiene la subcadena aa} \}$$

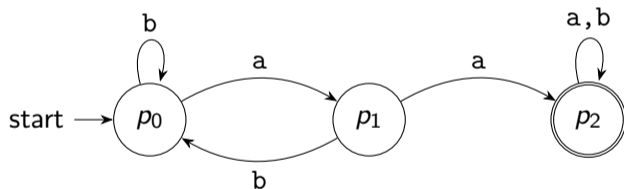
Ejemplos:

- **Acepta:** aa, baa, aab, bbaab
- **Rechaza:** ab, babab, bbb

Idea de diseño: “¿cuánto del patrón aa ya llevo visto?”

Diseño del AFD para L_2 (diagrama de estados)

Periodo 4/6



Significado de estados:

- p_0 : no he visto a reciente
- p_1 : el último símbolo fue a
- p_2 : ya apareció aa (**acepta**)

Nota: p_2 es absorbente: una vez encontrado aa, se mantiene aceptando.

Con el AFD de L_2 , responde (mentalmente o en tu cuaderno):

- ¿Acepta baab? (pista: sí contiene aa)
- ¿Acepta abab? (pista: no contiene aa)

Regla: si puedes justificar el **recorrido de estados**, puedes justificar la respuesta.

Objetivo del bloque: **simular paso a paso** y detectar errores de diseño.

- 1 Crea un autómata para L_1 (termina en ab).
- 2 Marca el estado de aceptación correctamente.
- 3 Ejecuta al menos 6 pruebas: 3 acepta / 3 rechaza.
- 4 Si falla una prueba, ajusta transiciones y vuelve a simular.

Salida del bloque: tu AFD guardado y una lista breve de pruebas usadas.

Antes de “dar por bueno” un AFD:

- ¿Cada estado tiene transición definida para a y para b ?
- ¿El estado inicial está claro?
- ¿Los estados de aceptación corresponden al lenguaje (no al dibujo)?
- ¿Probaste casos borde? (por ejemplo, ab , a , b , ϵ)

Consejo: prueba primero cadenas cortas; detectan errores más rápido.

Diseña un AFD para:

- (A) $L = \{w \in \{a, b\}^* : w \text{ termina en } ba\}$
- (B) $L = \{w \in \{a, b\}^* : w \text{ contiene la subcadena } bb\}$

Guarda tu autómata y prepara 5 pruebas (3 acepta / 2 rechaza).

- Hoy: qué es un **AFD**, cómo **diseñar** y cómo **verificar paso a paso**.
- Puente: mañana/sábado pasamos a práctica más larga y evidencia (AP1).
- Próximo: AFN y ϵ -AFN (cuando corresponda en el cronograma).

Recordatorio: el lenguaje es la meta; el autómata es el mecanismo.