2.1 Clasificación: Tipos de Autómatas Finitos

- **Autómata Finito Determinista (AFD):**
  - En cada paso, el autómata está en **exactamente un estado**.
  - Para cada símbolo leído, hay **una única transición posible**.
  - Ejemplo: Un candado digital que responde de manera clara y única a cada número presionado.
- **Autómata Finito No Determinista (AFN):**
  - El autómata puede estar en **varios estados al mismo tiempo** (como si explorara todos los caminos posibles en paralelo).
  - Para un símbolo leído, puede haber **varias transiciones posibles**, ninguna, o incluso transiciones "vacías" (ε).
  - Ejemplo: Un robot que ante una señal puede girar a la izquierda, derecha o quedarse quieto, explorando todas las opciones.

2.2 Autómata Finito Determinista (AFD)

Un **AFD** es una máquina que recibe cadenas de símbolos y decide si pertenecen a un **lenguaje regular** (responde "aceptada" o "rechazada").

Determinismo:
Desde cada estado y símbolo, **solo hay una transición**.
Nunca hay ambigüedad: siempre sabemos el siguiente paso.

Definición Formal

Un AFD se define como una 5-tupla:

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

- Q: Conjunto finito de **estados**
- Σ: Conjunto finito de **símbolos del alfabeto**
- δ: **Función de transición** ($Q \times \Sigma \to Q$; devuelve **un solo estado**)
- q_0: **Estado inicial** (pertenece a Q)
- F: Subconjunto de Q, **estados finales** (aceptación)

Ejemplo de AFD

Diseñar un autómata que reconoce las cadenas que tienen un número par de cadenas "01".

Visualización: - **Grafo:** Cada nodo es un estado, las flechas son transiciones etiquetadas por símbolos. - **Tabla de transiciones:** Estados en filas, símbolos en columnas. Marca con flecha el estado inicial y con * el final.

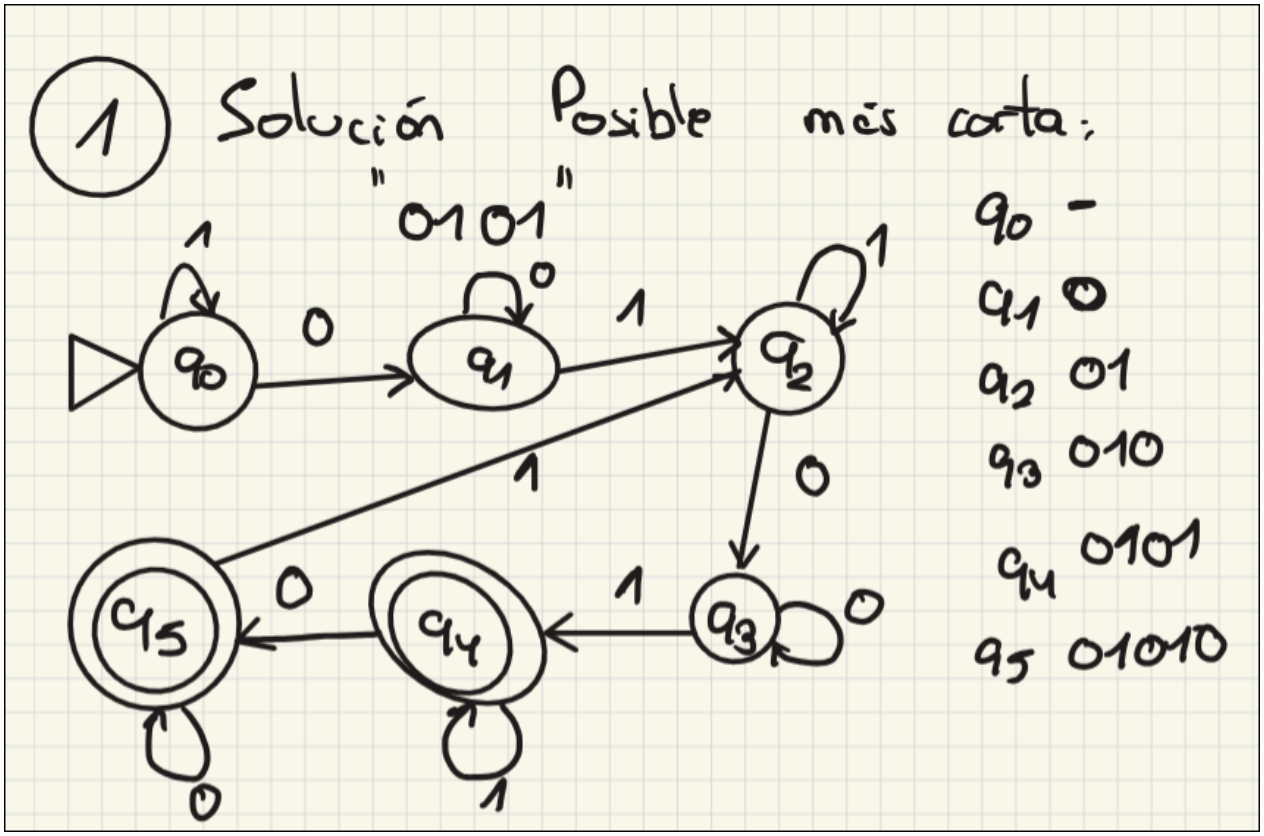


Figure 1: Pasted image 20251009161639.png

2.3 Autómata Finito No Determinista (AFN)

Un **AFN** también reconoce lenguajes regulares, pero su definición es más flexible:

- Desde un estado y símbolo, puede haber **cero, una o varias transiciones**.
- Puede tener **transiciones ε**, que permiten cambiar de estado sin consumir símbolo.

Esto significa que el autómata puede "probar" varios caminos al mismo tiempo, y acepta la cadena si **algún camino** termina en estado final.

Definición Formal

Un AFN se define como:

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

- Q: Conjunto finito de **estados**
- Σ: Alfabeto de símbolos
- δ: **Función de transición** ($Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$; devuelve **un conjunto de estados**)
- q_0: Estado inicial
- F: Estados finales

Ejemplo de AFN

Un autómata que reconoce las cadenas cuyo segundo símbolo empezando por la izquierda coincide con su segundo símbolo empezando por la derecha.

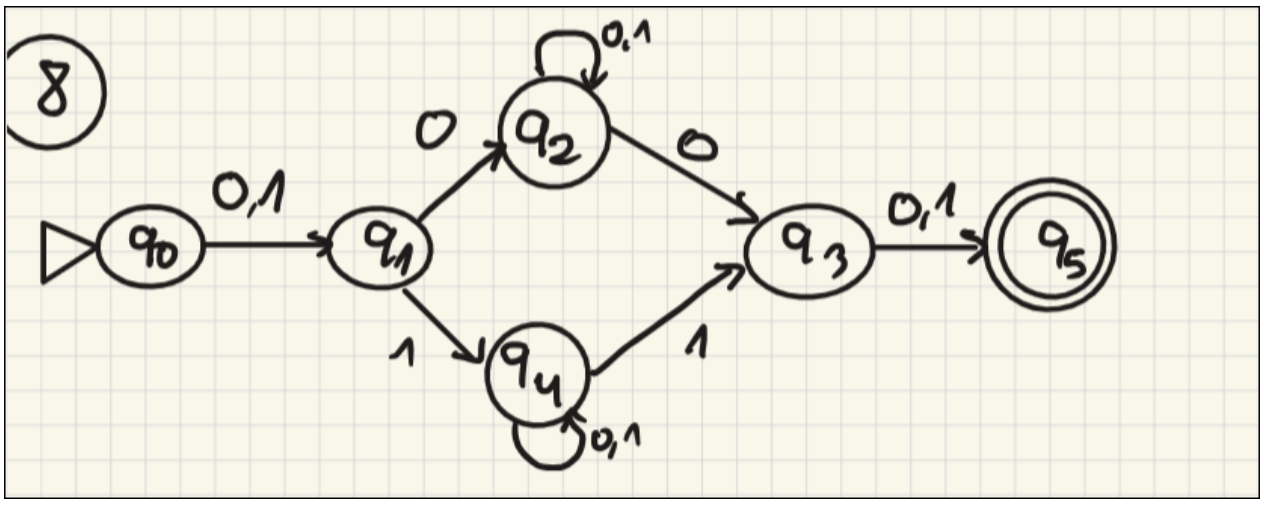


Figure 2: Pasted image 20251009161852.png

En el AFN, puedes estar en varios estados a la vez y elegir entre múltiples caminos.

AFN con Transiciones ε (AFN-ε)

Un **AFN-ε** permite transiciones vacías (ε):

- **Transición ε:** Cambiar de estado sin leer ningún símbolo.
- Usadas para simplificar el diseño de autómatas, pero **no aumentan el poder** (siguen reconociendo solo lenguajes regulares).

  [!Nota] Según el profesor, una transición vacía es una abstracción fundamental para poder construir los autómatas aunque no tenga un uso real, puesto que nunca se va a introducir ningún símbolo vacío al autómata. Son útiles para diseñar compiladores

ε se puede pensar en programación como la cadena vacía lo que sería equivalente a hacer por ejemplo en java:

```
String str = "";
```

Clausura ε

Cuando hay transiciones ε, usamos la **clausura ε** para saber, desde un estado, a cuántos estados puedo llegar usando solo transiciones ε (sin leer símbolos). - $clausura - \varepsilon(q)$: Todos los estados alcanzables desde q usando solo transiciones ε (incluido q mismo). - Para conjuntos de estados T, la clausura es la unión de las clausuras de cada estado en T.

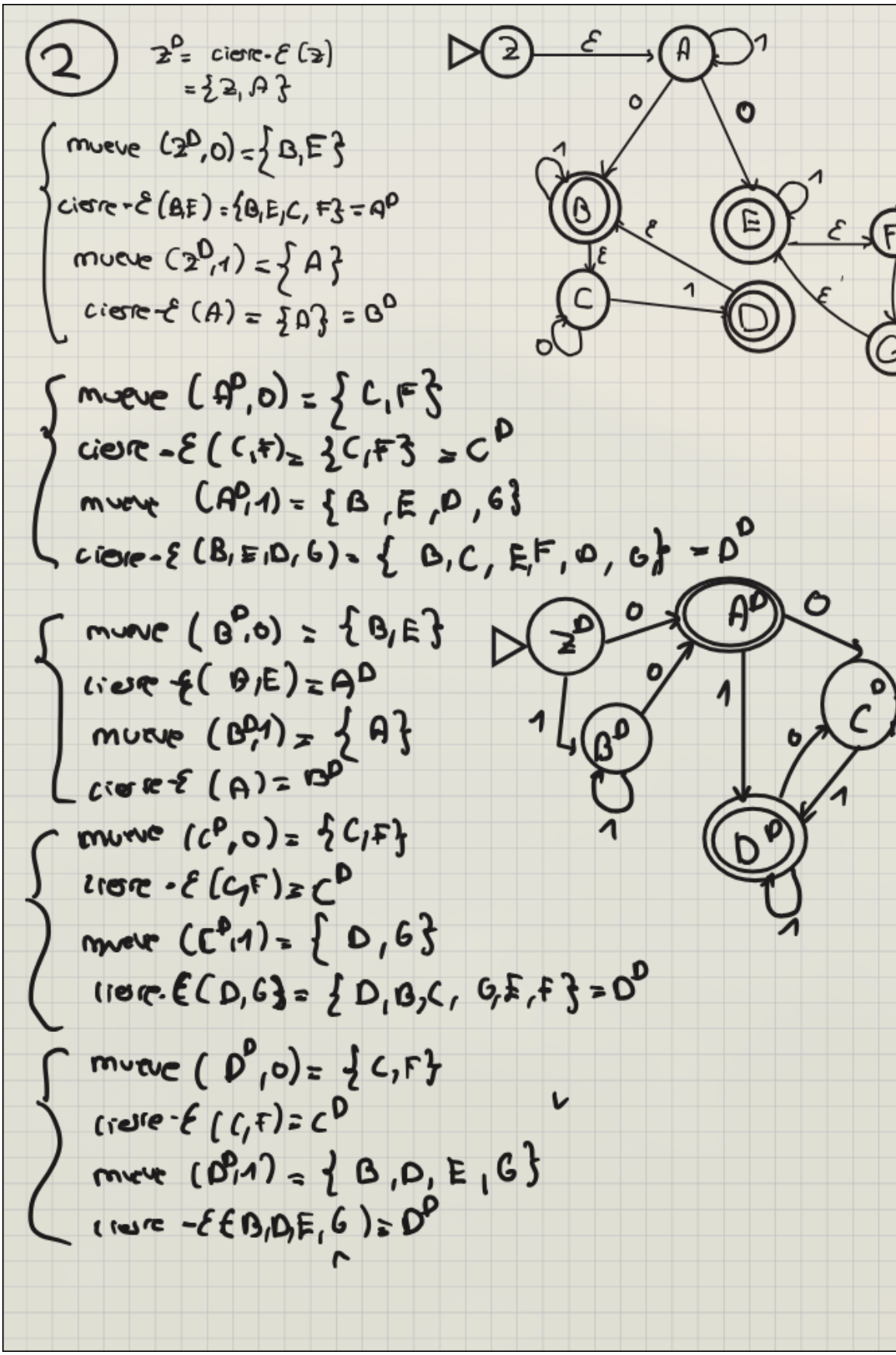
Ejemplo:
Si q_0 tiene una transición ε hacia q_1, entonces $clausura - \varepsilon(q_0)$= {$q_0$, q_1}.
Si además q_1 tiene una ε hacia q_2, entonces $clausura - \varepsilon(q_0)$= {$q_0$, q_1, q_2}.

2.4 Equivalencia y Conversión: AFN → AFD

Como el AFN puede estar en varios estados a la vez, en el AFD **cada estado representa un conjunto de estados del AFN**.

Pasos: 1. **Estado inicial del AFD**: La clausura ε del estado inicial del AFN ($S_0 = clausura - \varepsilon(\{q_0\})$). 2. **Transiciones**: Para cada conjunto de estados T del AFD y cada símbolo a, calcular: - $U = clausura - \varepsilon(mover(T, a))$ - donde $mover(T, a)$ es la unión de los destinos posibles al leer a desde cada $p \in T$ en el AFN. - U es un (posible nuevo) estado del AFD. 3. **Repetir**: Para cada nuevo conjunto de estados generado, repetir el proceso para todos los símbolos. 4. **Estados finales**: Todo conjunto T del AFD que contenga algún estado final del AFN es estado final del AFD.

Ejemplo práctico:



2.5 Minimización de AFD

1. **Partición inicial**: Separamos los estados en dos grupos: finales y no finales.
2. **Refinamiento**: Dividimos los grupos si, para algún símbolo, los estados del mismo grupo van a diferentes grupos.
3. **Repetir** hasta que no se puedan dividir más.
4. **Construcción**: Cada grupo final es un estado del autómata minimizado; las transiciones se definen por el comportamiento del grupo.

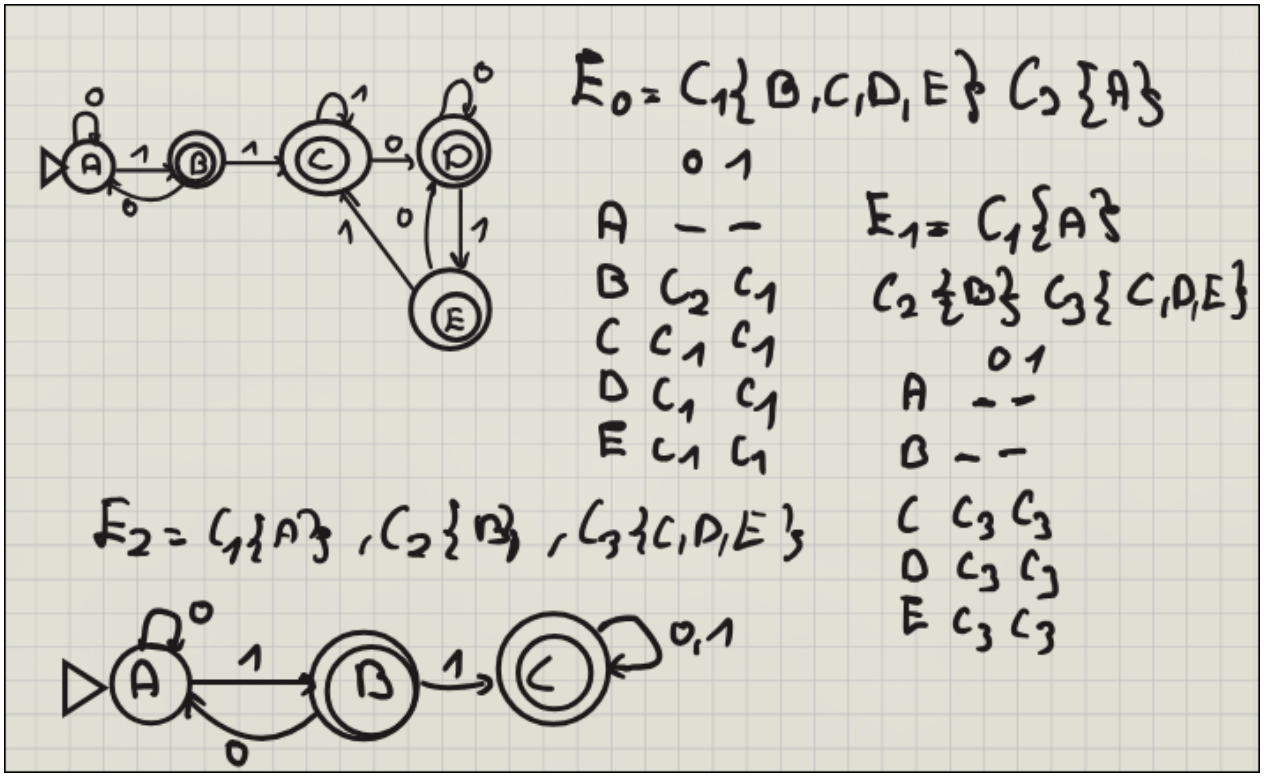


Figure 3: Pasted image 20251020130827.png