ACTIVIDAD 3.- AUTOMATA FINITO

QUIROZ TRUJILLO, OMAR ALEJANDRO

Traductores de Lenguaje II

Introducción

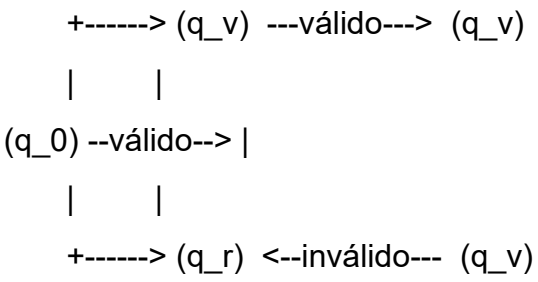
Es un sistema abstracto compuesto por un conjunto finito de estados y reglas de transición que determinan cómo se mueve entre estos estados en función de las entradas recibidas.

Existen dos tipos principales:

- **Autómata Finito Determinista (AFD)**: Para cada estado y símbolo de entrada, hay una única transición posible.
- **Autómata Finito No Determinista (AFND)**: Puede haber múltiples transiciones para una misma entrada desde un estado dado.

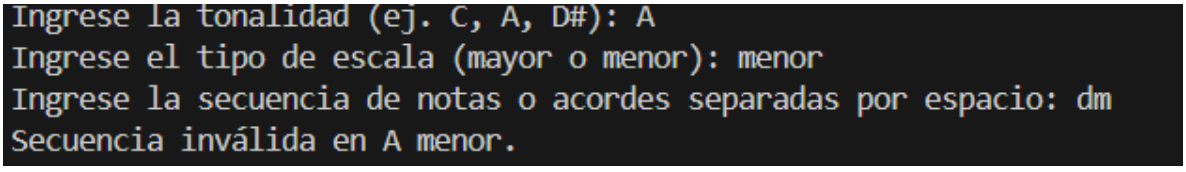
Su funcionamiento comienza en el estado inicial (Qo) y procesa una secuencia de símbolos de entrada uno por uno. En cada paso, sigue la transición definida por δ. Si al final de la secuencia se encuentra en un estado de aceptación (F), la entrada es válida; de lo contrario, es rechazada.

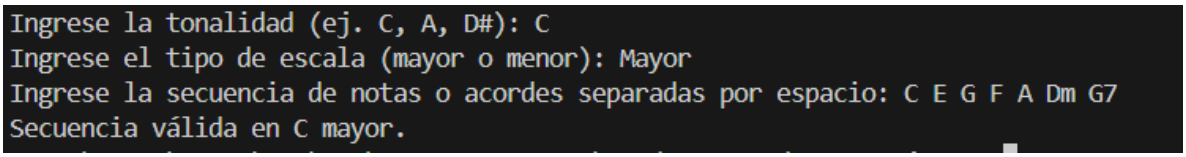
Diagrama del autómata



Tiene el propósito de validar si una secuencia de notas y acordes pertenece a una escala musical elegida por el usuario.

Ejemplos de ejecución





Código

```
class AFD_Escala:
    # Constructor de la clase
    def __init__(self, tonalidad, tipo):
        # Convierte la tonalidad a mayúsculas y el tipo de escala a minúsculas para uniformidad
        self.tonalidad = tonalidad.upper()
        self.tipo = tipo.lower()

        # Diccionario que almacena las escalas mayores y menores con sus respectivas notas y acordes
        self.escalas = {
            "mayor": {
                "C": {"notas": {"C", "D", "E", "F", "G", "A", "B"},
                      "acordes": {"C", "Dm", "Em", "F", "G", "Am", "Bdim", "G7"}},
                "A": {"notas": {"A", "B", "C#", "D", "E", "F#", "G#"},
                      "acordes": {"A", "Bm", "C#m", "D", "E", "F#m", "G#dim", "E7"}},
                "D#": {"notas": {"D#", "E#", "F##", "G#", "A#", "B#", "C##"},
                       "acordes": {"D#", "E#m", "F##m", "G#", "A#", "B#m", "C##dim", "A#7"}}
            },
            "menor": {
                "A": {"notas": {"A", "B", "C", "D", "E", "F", "G"},
                      "acordes": {"Am", "Bm-5", "C", "Dm", "Em", "F", "G", "E7"}},
                "D": {"notas": {"D", "E", "F", "G", "A", "Bb", "C"},
                      "acordes": {"Dm", "Em-5", "F", "Gm", "Am", "Bb", "C", "A7"}},
                "C": {"notas": {"C", "D", "Eb", "F", "G", "Ab", "Bb"},
                      "acordes": {"Cm", "Dm-5", "Eb", "Fm", "Gm", "Ab", "Bb", "G7"}}
```

```
            }
        }

        # Verifica si la tonalidad proporcionada está en la escala correspondiente
        if self.tonalidad not in self.escalas[self.tipo]:
            raise ValueError("Tonalidad no soportada")

    # Método para validar una secuencia de notas o acordes
    def validar_secuencia(self, secuencia):
        # Obtiene las notas y acordes válidos según la tonalidad y tipo de escala
        escala = self.escalas[self.tipo][self.tonalidad]
        notas_validas = escala["notas"]
        acordes_validos = escala["acordes"]

        # Recorre la secuencia ingresada y verifica si cada elemento es válido
        for item in secuencia.split():
            if item not in notas_validas and item not in acordes_validos:
                return f"Secuencia inválida en {self.tonalidad} {self.tipo}."

        # Si todos los elementos son válidos, devuelve un mensaje de éxito
        return f"Secuencia válida en {self.tonalidad} {self.tipo}."

# Ejemplo de uso
tonalidad = input("Ingrese la tonalidad (ej. C, A, D#): ")
tipo = input("Ingrese el tipo de escala (mayor o menor): ")

# Se crea un objeto de la clase AFD_Escala
afd = AFD_Escala(tonalidad, tipo)
```

```
# Se solicita al usuario ingresar una secuencia de notas o acordes
secuencia = input("Ingrese la secuencia de notas o acordes separadas por espacio: ")

# Se valida la secuencia ingresada y se muestra el resultado
resultado = afd.validar_secuencia(secuencia)
print(resultado)
```