**1. Finite State Machine (FSM)**

**Definitions**

A **Finite State Machine (FSM)** is a computational model consisting of a **finite number of states**, transitions between these states, and actions.

It is used to recognize patterns and model sequential logic in various applications like speech processing, lexical analysis, and text parsing.

**Formal Definition:**



An FSM is defined as a **5-tuple (Q, Σ, δ, q₀, F)** where:



* **Q**: Finite set of states.
* **Σ**: Finite set of input symbols (alphabet).



* **δ**: Transition function, mapping (Q × Σ → Q).



* **q₀**: Initial state (q₀ ∈ Q).
* **F**: Set of accepting (final) states (F ⊆ Q).



**Types of FSMs:**

1. **Deterministic Finite Automaton (DFA)**: One transition per state-input pair.
2. **Non-Deterministic Finite Automaton (NFA)**: Multiple transitions allowed per state-input pair.

**Example:**

A DFA recognizing the string "ab":



* **States**: {q₀, q₁, q₂}



* **Alphabet**: {a, b}
* **Transitions**:



* + q₀ → q₁ on 'a'
  + q₁ → q₂ on 'b'



* **Initial state**: q₀



* **Final state**: q₂

**2. Finite State Transducer (FST)**

A **Finite State Transducer (FST)** is an extension of an FSM that maps **input symbols to output symbols**, making it useful for applications like **morphological analysis, translation, and phonetic transcription**.

**Formal Definition:**

An FST is a **6-tuple (Q, Σ, Δ, δ, q₀, F)** where:

* **Q**: Finite set of states.
* **Σ**: Input alphabet.



* **Δ**: Output alphabet.



* **δ**: Transition function, mapping (Q × Σ → Q × Δ).
* **q₀**: Initial state.



* **F**: Set of final states.

**Example:**

An FST converting singular nouns to plural:

* **Input**: "cat" → **Output**: "cats"
* **Transitions**:
  + q₀ → q₁ on "cat/cat"
  + q₁ → q₂ on "ε/s" (adds "s" in plural form)

**Key Differences: FSM vs. FST**

| **Feature** | **Finite State Machine (FSM)** | **Finite State Transducer (FST)** |
| --- | --- | --- |
| **Purpose** | Accepts or rejects input strings | Transforms input to output |
| **Output** | No explicit output | Produces an output string |
| **Application** | Pattern recognition, lexical analysis | Morphological analysis, machine translation |
| **Example** | Recognizing valid words | Converting singular to plural |

**Examples of Finite State Machine (FSM) and Finite State Transducer (FST)**

**Example 1: Finite State Machine (FSM)**

**Scenario: A Simple Vending Machine**

A vending machine accepts coins (₹5 and ₹10) and dispenses a product when ₹15 is reached.

**FSM Components:**

* **States (Q)**: {q₀ (Start), q₁ (₹5 entered), q₂ (₹10 entered), q₃ (₹15 reached - Dispense)}
* **Input Alphabet (Σ)**: {₹5, ₹10}
* **Transition Function (δ)**:



* + q₀ → q₁ on ₹5



* + q₀ → q₂ on ₹10



* + q₁ → q₃ on ₹10 (Total ₹15)



* + q₂ → q₃ on ₹5 (Total ₹15)



* **Initial State**: q₀
* **Final State (F)**: q₃ (Product dispensed)

**State Transition Diagram:**

(Start)

q₀

/ \

₹5/ ₹10\

q₁ q₂

| \ |

| ₹10 ₹5

| \ /

q₃ (Dispense)

**Explanation:**

* The machine starts at q₀ and moves to q₁ when ₹5 is inserted.
* If ₹10 is inserted at q₀, it moves directly to q₂.
* When the total reaches ₹15 (q₃), the product is dispensed.

**Example 2: Finite State Transducer (FST)**

**Scenario: Converting Singular Nouns to Plural**

An **FST** is used to convert singular English nouns into their plural forms.

**FST Components:**

* **States (Q)**: {q₀ (Start), q₁ (Processing), q₂ (Final Output)}
* **Input Alphabet (Σ)**: {cat, dog, book}
* **Output Alphabet (Δ)**: {cats, dogs, books}
* **Transition Function (δ)**:
  + q₀ → q₁ on cat/cat
  + q₁ → q₂ on ε/s (appends "s" to form plural)
  + Same process for "dog" → "dogs", "book" → "books"
* **Initial State**: q₀
* **Final State (F)**: q₂

**State Transition Diagram:**

q₀

|

"cat/cat"

|

q₁

|

"ε/s" (adds "s")

|

q₂ (cats)

**Explanation:**

* The FST reads **"cat"** and outputs **"cat"**.
* Then, it applies the transformation **"ε → s"**, making it **"cats"**.
* This process generalizes to **dog → dogs**, **book → books**.

**Key Differences in Examples**

| **Feature** | **Finite State Machine (FSM) (Vending Machine)** | **Finite State Transducer (FST) (Pluralization)** |
| --- | --- | --- |
| **Input** | Coins (₹5, ₹10) | Singular noun |
| **Output** | Accept/reject + dispense product | Pluralized noun |
| **Transitions** | Moves between money collection states | Maps input to modified output |
| **Final Output** | Product dispensed | New word (e.g., "cats") |

**Additional Examples of Finite State Machine (FSM) and Finite State Transducer (FST)**

**Example 3: Finite State Machine (FSM) – Traffic Light System**

**Scenario: Traffic Light Control**

A traffic light system transitions between three states: **Red**, **Green**, and **Yellow**.

**FSM Components:**

* **States (Q)**: {Red, Green, Yellow}
* **Input Alphabet (Σ)**: {Timer Expiry}
* **Transition Function (δ)**:
  + Red → Green (on Timer Expiry)
  + Green → Yellow (on Timer Expiry)
  + Yellow → Red (on Timer Expiry)
* **Initial State**: Red
* **Final State**: The system loops infinitely, so it has no final state.

**State Transition Diagram:**

(Start)

↓

[Red] → [Green] → [Yellow] → [Red] (loop)

**Explanation:**

* The system **starts at Red**.
* When the **timer expires**, it **transitions to Green**.
* After another **timer expiry**, it **switches to Yellow**.
* Finally, it **returns to Red**.

**Example 4: Finite State Transducer (FST) – Spelling Correction**

**Scenario: Correcting a Typo**

An FST maps **incorrectly typed words** to **correct words**.

**FST Components:**

* **States (Q)**: {q₀ (Start), q₁ (Processing), q₂ (Final)}
* **Input Alphabet (Σ)**: {teh, recieve}
* **Output Alphabet (Δ)**: {the, receive}
* **Transition Function (δ)**:
  + q₀ → q₁ on teh/the
  + q₁ → q₂ (Final state)
  + q₀ → q₁ on recieve/receive
* **Initial State**: q₀
* **Final State (F)**: q₂

**State Transition Diagram:**

q₀

|

"teh/the"

|

q₁

|

"recieve/receive"

|

q₂ (Corrected Output)

**Explanation:**

* If a user types **"teh"**, the FST **corrects it to "the"**.
* If a user types **"recieve"**, the FST **corrects it to "receive"**.

**Example 5: FSM – Password Validator**

**Scenario: Checking a Strong Password**

A system verifies if a password follows these rules:

1. Must have at least **one uppercase letter**.
2. Must have at least **one digit**.
3. Must have at least **one special character (@, #, $)**.

**FSM Components:**

* **States (Q)**: {q₀ (Start), q₁ (Uppercase), q₂ (Digit), q₃ (Special Char), q₄ (Accepted)}
* **Input Alphabet (Σ)**: {A-Z, 0-9, @#$, others}
* **Transition Function (δ)**:
  + q₀ → q₁ on [A-Z]
  + q₁ → q₂ on [0-9]
  + q₂ → q₃ on [@#$]
  + q₃ → q₄ (Accepted)
* **Initial State**: q₀
* **Final State (F)**: q₄ (Password accepted)

**State Transition Diagram:**

q₀ (Start)

|

[A-Z] → q₁

|

[0-9] → q₂

|

[@#$] → q₃

|

q₄ (Accepted)

**Explanation:**

* If the user enters an **uppercase letter**, they move to q₁.
* If they enter a **digit**, they move to q₂.
* If they enter a **special character**, they move to q₃.
* If all conditions are met, they reach **q₄ (Password Accepted).**

**Example 6: FST – Number-to-Word Conversion**

**Scenario: Converting Numbers to Words**

An FST converts **digits** to **spoken words**.

**FST Components:**

* **States (Q)**: {q₀ (Start), q₁ (Processing), q₂ (Output)}
* **Input Alphabet (Σ)**: {0, 1, 2, ..., 9}



* **Output Alphabet (Δ)**: {zero, one, two, ..., nine}



* **Transition Function (δ)**:
  + q₀ → q₁ on 1/one
  + q₁ → q₂ (Final state)
  + q₀ → q₁ on 2/two
* **Initial State**: q₀
* **Final State (F)**: q₂

**State Transition Diagram:**

q₀

|

"1/one"

|

q₁

|

"2/two"

|

q₂ (Output)

**Explanation:**

* If the user inputs 1, the **FST converts it to "one"**.
* If the user inputs 2, the **FST converts it to "two"**.

**Example 7: FSM – Email Validation**

**Scenario: Checking a Valid Email Format**

A system ensures an email is in **valid format**: "username@domain.com"

**FSM Components:**

* **States (Q)**: {q₀ (Start), q₁ (Username), q₂ (At-symbol '@'), q₃ (Domain), q₄ (Dot '.'), q₅ (TLD), q₆ (Accepted)}
* **Input Alphabet (Σ)**: {a-z, A-Z, 0-9, @, ., com, org, net}
* **Transition Function (δ)**:
  + q₀ → q₁ on [a-zA-Z0-9]
  + q₁ → q₂ on @
  + q₂ → q₃ on [a-zA-Z]
  + q₃ → q₄ on . (dot)
  + q₄ → q₅ on com|org|net
  + q₅ → q₆ (Accepted)
* **Initial State**: q₀
* **Final State (F)**: q₆ (Valid Email)

**State Transition Diagram:**

q₀ (Start)

|

[a-z0-9] → q₁

|

'@' → q₂

|

[domain] → q₃

|

'.' → q₄

|

(com/org) → q₅

|

q₆ (Valid)

**Explanation:**

* If the email starts with **letters/numbers**, it moves to q₁.
* If @ is detected, it moves to q₂.
* If a **domain** is entered, it moves to q₃.
* If a **dot (.)** is entered, it moves to q₄.
* If **TLD (com/org/net)** is entered, it reaches q₆ (Valid Email).

**Summary**

| **FSM Examples** | **FST Examples** |
| --- | --- |
| **Traffic Light System** | **Pluralization (cat → cats)** |
| **Vending Machine** | **Spelling Correction (teh → the)** |
| **Password Validator** | **Number-to-Word (1 → one), …(9🡪nine)** |
| **Email Validation** | **Singular-to-Past Tense (run → ran)** |