Comprehensive & Simplified Finite Automata Notes

---

1. Finite State Automata (FSA) The word “automata” comes from a Greek word meaning “self-acting.” An automaton is an abstract, self-operating computing system that follows a fixed sequence of steps automatically. When an automaton has a finite number of states, it is called a Finite State Automaton (FSA). A finite automaton is a state machine that takes a string of symbols as input and changes its states according to those symbols. It is mainly used to recognize regular expressions. For every symbol in the input, the automaton moves from one state to another. If the automaton reaches a final (accepting) state after processing the entire input string, the string is accepted. Otherwise, it is rejected. A finite automaton takes one input string and produces one output: ✔️Accepted or ✘ Rejected.

Example: - Input string: "aab" - Alphabet Σ = {a, b} - The automaton starts at the initial state, reads symbols one by one, moves through intermediate states, and reaches the final state if accepted.

---

2. Finite Automata Model A finite automaton can be represented using an input tape and a finite control. The input tape is a linear tape divided into several cells, each containing one symbol from the input string. The input tape is read-only, meaning the automaton can only read symbols but cannot modify them. The finite control acts like the “brain” of the automaton and decides the next state based on the symbol it reads. The tape reader reads symbols from left to right, one at a time, and the automaton cannot move backward. This model demonstrates how a finite automaton processes a string step by step using simple rules.

Additional Explanation: - The finite control processes the input symbol and determines which state to move to next. - Each step in processing a symbol is deterministic in DFA or may have multiple possibilities in NFA. - The model highlights how computation progresses in a single forward pass over the input.

---

3. Finite Automata Construction To construct a finite automaton, we use a 5-tuple: 1. **States (Q):** Represented with circles; state names are written inside. 2. **Start State (q_0):** The initial state, indicated by an arrow pointing to it. 3. **Intermediate States:** States the automaton passes through to reach a final state, usually with at least two transitions. 4. **Final State (F):** Reached when the input string is successfully processed, represented with a double circle. 5. **Transition (δ):** Indicates the change from one state to another based on the input symbol.

Step-by-Step Construction: - Define all states including start, intermediate, and final states. - Determine the input alphabet Σ. - Draw transitions for each input symbol according to δ. - Mark the start state and double-circle the final states. - Ensure all symbols of the alphabet are handled in DFA for determinism.

---

4. Deterministic Finite Automata (DFA) Definition: A DFA has exactly one path for each input symbol from a given state and cannot have null moves (ε-moves). Represented as a 5-tuple: (Q, Σ, δ, q_0, F).

Operation: - Start in initial state q_0. - Read input string one symbol at a time from left to right. - Apply transition function δ to move to next state (e.g., $\delta(q_0, a_1) \rightarrow q_1$). - Continue until all symbols are read. - If the DFA ends in a final state, the string is accepted; otherwise, rejected.

Key Points: - No null transitions. - Deterministic computation with one unique path per input. - Useful in compilers, lexical analyzers, and exact pattern recognition. - Processes input in a single forward pass.

Output:✔️ Accepted / ✘ Rejected

Example: - Alphabet Σ = {a, b} - DFA designed to accept strings ending with 'a'. - Input string: "ba" - DFA moves through $q_0 \rightarrow q_1 \rightarrow$ final state; string accepted.

---

5. Language Recognition Language recognition determines whether a string belongs to a specific language. A language is a set of strings over an alphabet Σ. A finite automaton M recognizes a string w if after processing the string, it ends in a final (accepting) state.

Example: - Alphabet Σ = {a, b} - Language L = strings ending with 'a' - Accepted: "a", "ba", "bba", "aa" ✔️- Rejected: "b", "ab", "bb" ✘ - Stepwise: The automaton reads each symbol, changes state according to δ, and accepts if it reaches the final state.

---

6. Non-Deterministic Finite Automata (NFA) Definition: An NFA can have multiple possible transitions for an input symbol from a state and allows null moves (ε-moves). Represented as a 5-tuple: $(Q, \Sigma, \delta, q_0, F)$.

Operation: - Start in initial state q_0. - Read input symbols from left to right. - Move to one or more next states for each symbol according to δ. - Can move to a new state without reading a symbol (ε-move). - String is accepted if any computation path reaches a final state; otherwise rejected.

Example: - Alphabet Σ = {a, b} - NFA designed to accept strings ending with 'a'. - Input string: "ba" - Multiple possible paths are evaluated; if any path reaches the final state, string is accepted.

Additional Notes: - Finite automata are foundational for formal languages, compiler design, and pattern recognition. - Diagrams help visualize states, transitions, and final states. - FSAs process input strings systematically and predictably. - DFAs are deterministic with unique paths; NFAs can have multiple paths and ε-moves.