FINITE AUTOMATA

Chap. 2

**Summary**

* The simple automaton, a finite state accepter:
* a finite set of internal states and no other memory.
* It process strings and either accepts or reject them.
* A simple pattern recognition mechanism.
* Deterministic Finite Automaton/Accepter(DFA)
* Deterministic: the automaton has only one transition to a next state per a symbol at one time.
* Regular language
* Nondeterministic Finite Automaton/Accepter(NFA)
* Nondeterministic: several transitions for a choice of next state.
* NFA explores all choices and makes no decision until all options have been analyzed.
* NFA simplifies the solution of many problems.
* Equivalence of DFA & NFA

**Learning Objectives**

* Describe the components of a Deterministic Finite Automata(DFA).
* State whether an input string is accepted by a DFA.
* Describe the language accepted by a DFA.
* Construct a DFA to accept a specific language.
* Show that a particular language is regular.
* Describe the differences between DFA and NFA.
* State whether an input string is accepted by a NFA.
* Construct a NFA to accept a specific language.
* Transform an arbitrary NFA to an equivalent DFA.

**Deterministic Finite Automata**

* Formal Definition 2.1: A Deterministic Finite Automata, DFA, (or accepter, recognizer)is defined by the quintuple M = (Q, Σ, δ, q0, F) where Q : a finite set of internal statesΣ: a set of symbols, called the input alphabetδ∶ 𝑄ൈΣ→ 𝑄‐‐a transition functionq0 ∈Q : the initial state F (⊆Q) : a set of the final states.
* Example 2.1: Consider the DFA Q = { q0,q1,q2 },Σ= { 0, 1 }, F = { q1}where the transition function is given by{δ(q0, 0) = q0, δ(q0, 1) = q1, δ(q1, 0) = q0δ(q1, 1) = q2 ,δ(q2, 0) = q2,δ(q2, 1) = q1 }.

**Transition Diagram/Graph**

* A transition function of DFA can be visualized with a Transition Diagram.
* Example 2.1: M = (Q, Σ, δ, q0, F) where Q = { q0,q1,q2 }, Σ= { 0, 1 }, F = { q1}{δ(q0, 0) = q0, δ(q0, 1) = q1, δ(q1, 0) = q0δ(q1, 1) = q2 ,δ(q2, 0) = q2, δ(q2, 1) = q1 }.

**Processing Input with a DFA**

* A DFA starts by processing the leftmost input symbol with its control in state q 0. The transition function determines the next state, based on current state and input symbol.
* The DFA continues processing input symbols until the end of the input string is reached.
* The input string is accepted if the automaton is in a final state after the last symbol is processed. Otherwise, the string is rejected.
* For example, the DFA in Ex. 2.1 accepts the string 111 but rejects the string 110.
* The Language Accepted by a DFA
* For a given DFA, the extended transition function δ\* accepts a DFA state and an input string as input. The value of the function is the state of the automaton after the string is processed: δ∗:𝑄ൈΣ∗→ 𝑄, s.t.•δ∗(q, λ) = q,•δ∗(q, wa) = δ(δ∗(q, w), a), w∈Σ∗, a∈Σ•Sample values of δ\* for the DFA in Ex. 2.1,δ\*(q0, 1001) = q1 (∈F), δ\*(q1, 000) = q0 (∉F)
* The Language Accepted by a DFA (cont.)
* Def 2.2: The language accepted by a DFAM= (Q, Σ, δ, q0, F) is the set of all strings on Σaccepted by M, i.e. the set of all strings w whose transition results in a final state. Formally, L(M) = {w∈Σ\*| δ\*(q 0 , w) ∈F}.•Note: The language rejected by a DFA M is𝐿ሺ𝑀ሻൌሼ𝑤∈Σ\*| δ\*(q0 , w) ∉F}
* Example: DFA (Acceptor, Recognizer)
* Example 2.2: a DFA to accept the set of all strings on Σ={a, b}, consisting of an arbitrary number of a’s, followed by a single b, i.e. L = {anb| n≥0 }. •Note that q2is a trap state.
* Example: DFA (Acceptor, Recognizer)
* Example 2.3: a DFA to accept the set of all strings on {a, b} that start with the prefix ab, i.e. { abw| w∈{a, b}\* }
* Note that q3is a trap state.
* Example: DFA (Acceptor, Recognizer)
* Example 2.4: a DFA M’ to accept all strings on Σ={0, 1}, except those containing the substring 001. i.e. L(M) = { v001w| δ\*(q0 , v001w) ∈F ∀v, w ∈{0,1}\* } ◊L(M’) = L(M)= { u| δ\*(q0 , v001w ) ∉F, ∀u, v, w ∈{0,1}\* }
* Note that any state except a state ‘q001’ is a final state.
* Regular Languages
* Finite automata accept a family of languages collectively known as regular languages. •Def. 2.3: A language Lis regular if and only if there exists a DFA, M, that accepts L, i.e. L= L(M).•To show that a language is regular, one must construct a DFA to accept it.
* Example: Show that L = {(ab)na| n> 0} is regular.◊Construct a DFA, M, that accepts L, i.e. L= L(M).L= {aba, ababa, abababa, ......}
* Regular languages have wide applicability in problems that involve scanning input strings in search of specific patterns.
* Example: Regular Languages
* Example 2.6:Show the language L2= LL is regular,L2={aw1aaw2a | w1,w2∈{a,b}\*}.•Example 2.5:Show that the language L={awa| w∈{a,b}\*} is regular.
* Nondeterministic Finite Automaton (NFA)•An automaton is nondeterministicif it has a choice of actions for given inputs.
* Def 2.4: A Nondeterministic Finite Automata, NFA, (or accepter, recognizer)is defined by the quintuple M = (Q, Σ, δ, q0, F) where Q, Σ, q0, F are defined as for DFA, butδ: 𝑄ൈΣ∪λ→𝟐𝑸‐‐a transition function•Basic Differences between DFA and NFA:1. In an NFA, a transition of (state, symbol) may lead to several states simultaneously.2. If a transition is labeled with the empty string (λሻas its input symbol, the NFA may change states without consuming input (i.e. with λ).3. An NFA may have undefined transitions.
* Example: Nondeterministic FA•Example 2.7: a Nondeterministic FA in which there are two transitions labeled ‘a’out of state q0:δ(q0 , a) = { q1, q4 }