ToC

Monday, September 4, 2023 11:14 PM

Theory of computation

Automata theory (also known as **Theory of Computation**) is a theoretical branch of Computer Science and Mathematics

is the branch that deals with how efficiently problems can be solved on a model of computation, using an algorithm

deals with the logic of computation with respect to simple machines, referred to as automata

Automata* enables the scientists to understand how machines compute the functions and solve problems

"What are the fundamental capabilities and limitations of computers?

- Automaton=an abstract computing device
  - a device need not even be a physical hardware

Central Concepts of Automata Theory

- Symbols ---- a, b, 0, 1,2 etc
- Alphabet --- a set of symbols
- Strings --- a sequence of symbols from an alphabet
- Language --- a set of strings from the same alphabet

Alphabets

An alphabet is a finite, nonempty set of symbols.

- Conventional notation --- Σ
- The term "symbol" is usually undefined.
  - Examples ---
    - Binary alphabet $\Sigma = \{0, 1\}$.
    - $\Sigma = \{a, b, ..., z\}$ ...

Strings

A string (or word) is a finite sequence of symbols from an alphabet.

- Example ---
  - 1011 is a string from alphabet $\Sigma = \{0, 1\}$
- Empty string ε --- a string with zero occurrences of symbols
- Length $|w|$ of string w --- the number of positions for symbols in w
  - Examples --- $|0111|=4$, $|\varepsilon|=0$, ...

Languages

Languages

- More examples of languages ---
  - The set of all strings of n 0's followed by n 1's for $n \geq 0$:

    $\{\varepsilon, 01, 0011, 000111, \ldots\}$
  - Σ^* is an infinite language for any alphabet Σ.
  - ϕ = the empty language (not the empty string ε) is a language over any alphabet.
  - $\{\varepsilon\}$ is a language over any alphabet (consisting of only one string, the empty string ε).

Extended Transition Function

We describe the effect of a string of inputs on a DFA by extending $\hat{\delta}$ to a state and a string.

Induction on length of string.

Basis: $\hat{\delta}(q, \epsilon) = q$

Suppose w is a string where w=xa.

w=1101 is broken into x=110 and a=1.

Induction: $\hat{\delta}(q,w) = \delta(\hat{\delta}(q,x),a)=r$

$\hat{\delta}(q,w) = \delta(\hat{\delta}(q,x), a) = r$

x a

	0	1
A	A	B
B	A	C
C	C	C

$\hat{\delta}(B,011) = \delta(\hat{\delta}(B,01),1) = \delta(\delta(\hat{\delta}(B,0),1),1) =$

$\delta(\delta(A,1),1) = \delta(B,1) = C$

	0	1
A	A	B
B	A	C
C	C	C

$w = 011$

$$\hat{\delta}(B, 011)$$

$$= \delta(\hat{\delta}(B, 01), 1)$$

$$= \delta(\delta(\hat{\delta}(B, 0), 1), 1)$$

$$\delta(A, 1)$$

$$\delta(B, 1) = C$$

Extended transition function for NFA

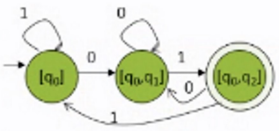
- Basis: $\hat{\delta}(q, \varepsilon) = \{q\}$

Induction: Suppose w is a string where $w = xa$.

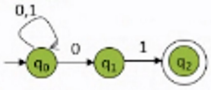
- Let $\hat{\delta}(q, x) = \{p_1, p_2 \ldots, p_k\}$
- $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, r_3, \ldots, r_m\}$
- $\hat{\delta}(q, w) = \{r_1, r_2, r_3, \ldots r_m\}$

NFA to DFA: Repeating the example using *LAZY CREATION*

$L = \{w \mid w \text{ ends in } 01\}$



NFA:



δ_N	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

DFA:

δ_D	0	1
$[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$*[q_0, q_2]$	$[q_0, q_1]$	$[q_0]$

Main Idea:
Introduce states as you go (on a need basis)

FA with ε-Transitions

- We can allow explicit ε-transitions in finite automata
  - i.e., a transition from one state to another state without consuming any additional input symbol
  - Explicit ε-transitions between different states introduce non-determinism.
  - Makes it easier sometimes to construct NFAs
  - This means that a transition is allowed to occur without reading in a symbol.

***Definition:* ε-NFAs are those NFAs with at least one explicit ε-transition defined.**

- ε -NFAs have one more column in their transition table

Transition function δ is now a function that takes as arguments:

- A state in Q and
- A member of $\Sigma \cup \{\varepsilon\}$; that is, an input symbol or the symbol ε. We require that ε not be a symbol of the alphabet Σ to avoid any confusion.

ε-Transitions

Use of e-transitions

We allow the automaton to accept the empty string e.

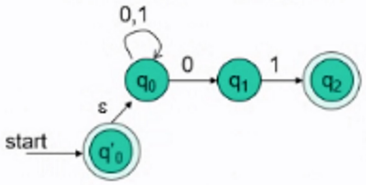
This means that a transition is allowed to occur without reading in a symbol.

The resulting NFA is called e-NFA.

It adds "programming (design) convenience" (more intuitive for use in designing FA's)

Example #2: ε-NFA..

L = {w | w is empty, or if non-empty will end in 01}

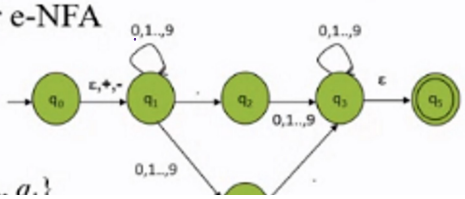


δ_E	0	1	ε-closure	
→*q'_0	Ø	Ø	$\{q'_0, q_0\}$	ECLOSE(q'_0)
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$	ECLOSE(q_0)
q_1	Ø	$\{q_2\}$	$\{q_1\}$	ECLOSE(q_1)
*q_2	Ø	Ø	$\{q_2\}$	ECLOSE(q_2)

- ε-closure of a state q, ***ECLOSE(q)***, is the set of all states (including itself) that can be *reached* from q by repeatedly making an arbitrary number of ε-transitions.

FA with ε-transition

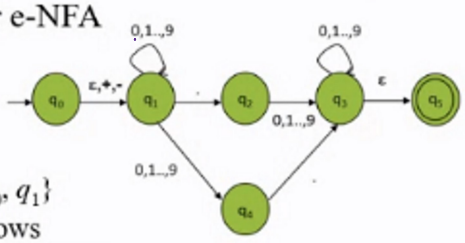
Computing $\hat{\delta}(q_0, 5.6)$ for e-NFA



- $\hat{\delta}(q_0, e) = \text{ECLOSE}(q_0) = \{q_0, q_1\}$

FA with ε-transition

Computing $\hat{\delta}(q_0, 5.6)$ for e-NFA



- $\hat{\delta}(q_0, e) = \text{ECLOSE}(q_0) = \{q_0, q_1\}$
- Compute $\hat{\delta}(q_0, 5)$ as follows

1. $\hat{\delta}(q_0, 5) = (q_0, e5) = \text{ECLOSE}(\delta(q_0, 5) \cup \delta(q_1, 5)) = \{q_1, q_4\}$
2. ECLOSE the result of step (1)
   $= \text{ECLOSE}(\{q_1, q_4\}) = \text{ECLOSE}(\{q_1\}) \cup \text{ECLOSE}(\{q_4\})$
   $= \{q_1, q_4\}$

- Compute $\hat{\delta}(q_0, 5.)$
- Compute $\hat{\delta}(q_0, 5.6)$

19

$\{\ \}$

$\hat{\delta}(q_0, 5.) = \text{ECLOSE}(\delta(q_1, .) \cup \delta(q_4, .))$

$= \text{ECLOSE}(q_2, q_3)$

$= \{q_2, q_3, q_5\}$

$\hat{\delta}(q_0, 5.6) = \text{ECLOSE}(\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6))$

$= \text{ECLOSE}(q_3)$,

$= \{q_3\ q_5\}$

RE's: Definition

Basis 1: If a is any symbol, then **a** is a RE, and L(**a**) = {a}.
- Note: {a} is the language containing one string, and that string is of length 1.

Basis 2: ϵ is a RE, and L(ϵ) = {ϵ}.

Basis 3: $\emptyset$ is a RE, and L($\emptyset$) = $\emptyset$.

RE's: Definition – (2)

Induction 1: If E_1 and E_2 are regular expressions, then E_1+E_2 is a regular expression, and $L(E_1+E_2) = L(E_1) \cup L(E_2)$.

Induction 2: If E_1 and E_2 are regular expressions, then E_1E_2 is a regular expression, and $L(E_1E_2) = L(E_1)L(E_2)$.

Concatenation : the set of strings wx such that w Is in $L(E_1)$ and x is in $L(E_2)$.

RE's: Definition – (3)

Induction 3: If E is a RE, then E* is a RE, and L(E*) = (L(E))*.

Closure, or "Kleene closure" = set of strings $w_1w_2...w_n$, for some $n \geq 0$, where each w_i is in L(E).

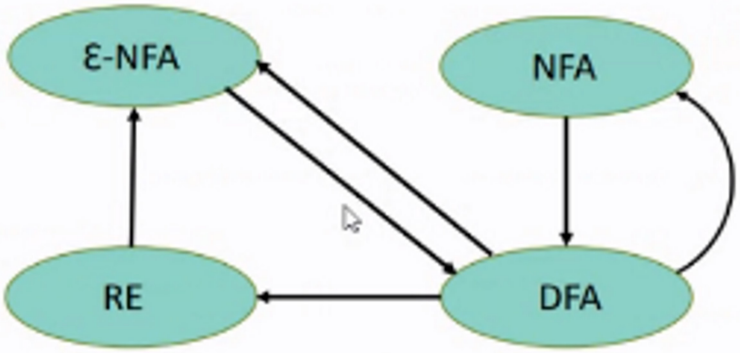
Note: when n=0, the string is ϵ.

Precedence of Operators

Parentheses may be used wherever needed to influence the grouping of operators.

Order of precedence is * (highest), then concatenation, then + (lowest).

From DFA to RE



$R^k_{i,j}$

Recursive step: for each k, we can build $R^k_{i,j}$ as follows:

$$R^k_{i,j} = R^{k-1}_{i,j} + R^{k-1}_{i,k} (R^{k-1}_{k,k})^* R^{k-1}_{k,j}$$

Intuition: since the accepting sequence contains one or more visits to state k, break the path into pieces that

- first goes from i to its first k-visit ($R^{k-1}_{i,k}$)
- followed by zero or more revisits to k ($R^{k-1}_{k,k}$)
- followed by a path from k to j ($R^{k-1}_{k,j}$)

Pumping Lemma for Regular Languages

Let L be a regular language

Then *there exists* some constant **N** such that *for every* string $w \in L$ s.t. $|w| \geq N$, *there exists* a way to break w into three parts, $w=xyz$, such that:

1. $y \neq \varepsilon$
2. $|xy| \leq N$
3. For all $k \geq 0$, all strings of the form $xy^kz \in L$

This property should hold for all regular languages.

Definition: N is called the "Pumping Lemma Constant"

8

$0^n 1^n$

if $n=3$, $w = 000111$

x y z

$|xy| \leq 3$

$\forall_{k \geq 0} \quad xy^k z \in L$

$K=0 \quad w = xz = 0111 \notin L$

$K=1 \quad w = xyz = 000111 \in L$

$K=2 \quad w = xyyz = 00000111 \notin L$

$0^n 1^n \rightarrow w \rightarrow xyz \;, |xy| \leq n$

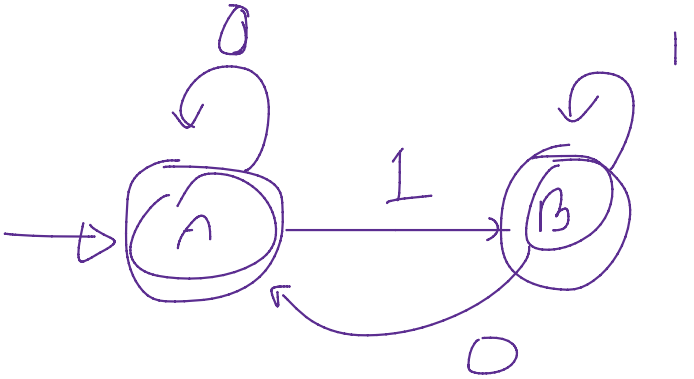
$w = \boxed{0^* 1^*} = \underbrace{00}_{x}\underbrace{1}_{y}\underbrace{11}_{z}$

$|xy| \leq \boxed{2}$

$\forall_{k \geq 0} \quad xy^k z \in L$

$k = 0 \quad w = xz = 0111 \in L$

$k = 2 \quad w = xyyz = \boxed{00111}$



$w = 00111$

$|w| = 5$

$n = 2 \rightarrow$ no of states in DFA

$|w| \geq n$