Intro to Analysis of Algorithms
Computational Foundations
Section 9.3
Chapter 9

Michael Soltys

CSU Channel Islands

[Ed: 4th, last updated: September 16, 2025]

Section 9.3:
Regular languages

Deterministic Finite Automaton (DFA)

$A = (Q, \Sigma, \delta, q_0, F)$

- Finite set of states Q
- Finite set of input symbols Σ
- Transition fn $\delta : Q \times \Sigma \longrightarrow Q$; given $q \in Q, a \in \Sigma$, $\delta(q, a) = p \in Q$
- Start state q_0
- A set of final (accepting) states.

To see whether A accepts a string w, we "run" A on $w = a_1 a_2 \dots a_n$ as follows:

$\delta(q_0, a_1) = q_1$, $\delta(q_1, a_2) = q_2$, until $\delta(q_{n-1}, a_n) = q_n$.

Accept iff $q_n \in F$.

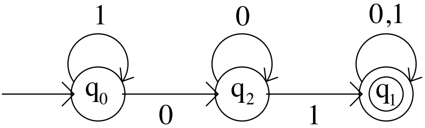
John von Neumann

Consider $L = \{w|\ w \text{ is of the form } x01y \in \Sigma^*\ \}$ where $\Sigma = \{0,1\}$.

We want to specify a DFA $A = (Q, \Sigma, \delta, q_0, F)$ that accepts all and only the strings in L.

$\Sigma = \{0,1\}$, $Q = \{q_0, q_1, q_2\}$, and $F = \{q_1\}$.

Transition diagram



Transition table

	0	1
q_0	q_2	q_0
q_1	q_1	q_1
q_2	q_2	q_1

Extended Transition Function (ETF) given δ, its ETF is $\hat{\delta}$ defined inductively:

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

Induction Step: if $w = xa$, $w, x \in \Sigma^*$ and $a \in \Sigma$, then

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

Thus: $\hat{\delta} : Q \times \Sigma^* \longrightarrow Q$.

$w \in L(A) \iff \hat{\delta}(q_0, w) \in F$

Here $L(A)$ is the set of all those strings (and only those) which are accepted by A.

Language of a DFA: $L(A) = \{w | \hat{\delta}(q_0, w) \in F\}$

Note that

- A is a *syntactic* object
- while $L(A)$ is a *semantic* object

Thus L is a function that assigns a *meaning* or *interpretation* to a syntactic object.

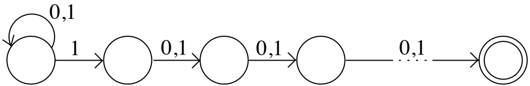
Regular Languages: L is *regular* iff there exists a DFA A such that $L = L(A)$.

Nondeterministic Finite Automata (NFA)

The transition function δ becomes a transition relation, i.e., $\delta \subseteq Q \times \Sigma \times Q$, i.e., on the same pair (q, a) there may be more than one possible new state (or none).

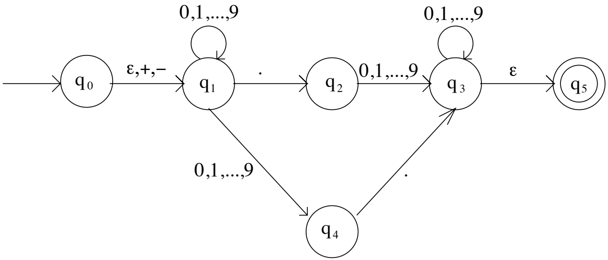
Equivalently, we can look at δ as $\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)$, where $\mathcal{P}(Q)$ is the power set of Q.

$L_n = \{w|\ n\text{-th symbol from the end is } 1\ \}$
What is an NFA for L_n



At least how many states does any DFA recognizing L_n require?

NFA with ε transitions: ε-NFA: $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow \mathcal{P}(Q)$



To define $\hat{\delta}$ for ε-NFAs we need the concept of ε-closure.

Given q, ε-close(q) is the set of all states p which are reachable from q by following arrows labeled by ε.

Formally, $q \in \varepsilon\text{-close}(q)$, and if $p \in \varepsilon\text{-close}(q)$, and $p \xrightarrow{\varepsilon} r$, then $r \in \varepsilon\text{-close}(q)$.

$\hat{\delta}(q, \varepsilon) = \varepsilon\text{-close}(q)$

Suppose $w = xa$, $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_n\}$,
and $\cup_{i=1}^{n} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$,
then

$$\hat{\delta}(q, w) = \cup_{i=1}^{m} \varepsilon\text{-close}(r_i)$$

Theorem: DFAs and ε-NFAs are equivalent.

Proof: Slightly modified subset construction.

$q_0^D = \varepsilon\text{-close}(\{q_0^N\})$

$\delta_D(R, a) = \cup_{r \in R} \varepsilon\text{-close}(\delta_N(r, a))$

Given a set of states S, its ε-closure is the union of the ε-closures of its members.

The states of D are those subsets $S \subseteq Q_N$ which are equal to their ε-closures.

Corollary: A language is regular
$\iff$ it is recognized by some DFA
$\iff$ it is recognized by some NFA
$\iff$ it is recognized by some ε-NFA

Union: $L \cup M = \{w | w \in L \text{ or } w \in M\}$
Concatenation: $LM = \{xy | x \in L \text{ and } y \in M\}$
Star (or closure): $L^* = \{w | w = x_1 x_2 \ldots x_n \text{ and } x_i \in L\}$

Regular Expressions

Basis Case: $a \in \Sigma, \varepsilon, \emptyset$

Induction Step: If E, F are regular expressions, the so are $E + F, EF, (E)^*, (E)$.

What are $L(a), L(\varepsilon), L(\emptyset), L(E+F), L(EF), L(E^*)$?

Ex. Give a reg exp for the set of strings of 0s and 1s not containing 101 as a substring:

$$(\varepsilon + 0)(1^* + 00^*0)^*(\varepsilon + 0)$$

Theorem: A language is regular iff it is given by some regular expression.

Proof: reg exp $\Longrightarrow$ ε-NFA & DFA $\Longrightarrow$ reg exp

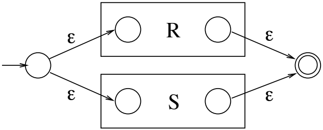
[$\Longrightarrow$]

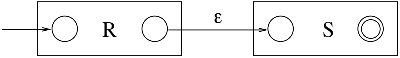
Use structural induction to convert R to an ε-NFA with 3 properties:

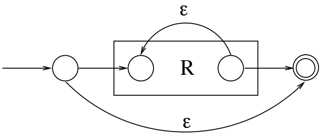
1. Exactly one accepting state
2. No arrow into the initial state
3. No arrow out of the accepting state

Basis Case: $\varepsilon, \emptyset, a \in \Sigma$

Induction Step: $R + S, RS, R^*, (R)$







[$\Longleftarrow$] Convert DFA to reg exp.

Method 1

Suppose A has n states. $R_{ij}^{(k)}$ denotes the reg exp whose language is the set of strings w such that:

> *w takes A from state i to state j with all intermediate states $\leq k$*

What is R such that $L(R) = L(A)$?

$R = R_{1j_1}^{(n)} + R_{1j_2}^{(n)} + \cdots + R_{1j_k}^{(n)}$ where $F = \{j_1, j_2, \ldots, j_k\}$

Build $R_{ij}^{(k)}$ by induction on k.

Basis Case: $k = 0$, $R_{ij}^{(0)} = x + a_1 + a_2 + \cdots + a_k$ where $i \xrightarrow{a_l} j$ and $x = \emptyset$ if $i \neq j$ and $x = \varepsilon$ if $i = j$

Induction Step: $k > 0$

$$R_{ij}^{(k)} = \underbrace{R_{ij}^{(k-1)}}_{\text{path does not visit } k} + \underbrace{R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}}_{\text{visits } k \text{ at least once}}$$

Method 2: DFA $\Longrightarrow$ Gε-NFA $\Longrightarrow$ Reg Exp

Generalized ε-NFA:

$$\delta : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathcal{R}$$

where the start and accept states are unique.

G accepts $w = w_1 w_2 \ldots w_n$, $\underline{w_i \in \Sigma^*}$, if there exists a sequence of states

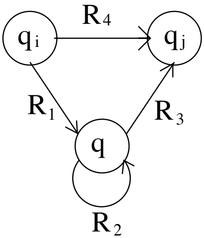
$$q_0 = q_{\text{start}}, q_1, \ldots, q_n = q_{\text{accept}}$$

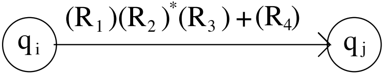
such that for all i, $w_i \in L(R_i)$ where $R_i = \delta(q_{i-1}, q_i)$.

When translating from DFA to Gε-NFA, if there is no arrow $i \longrightarrow j$, we label it with $\emptyset$.

For each i, we label the self-loop with ε.

Eliminate states from G until left with just $q_{\text{start}} \xrightarrow{R} q_{\text{accept}}$:





Algebraic Laws for Reg Exps

$L+M=M+L$ (commutativity of $+$)
$(L+M)+N=L+(M+N)$ (associativity of $+$)
$(LM)N=L(MN)$ (associativity of concatenation)
$LM=ML$?

$\emptyset+L=L+\emptyset=L$ ($\emptyset$ identity for $+$)
$\varepsilon L=L\varepsilon=L$ (ε identity for concatenation)
$\emptyset L=L\emptyset=\emptyset$ ($\emptyset$ annihilator for concatenation)

$L(M+N)=LM+LN$ (left-distributivity)
$(M+N)L=ML+NL$ (right-distributivity)

$L + L = L$ (idempotent law for union)

Laws with closure:

$(L^*)^* = L^*$
$\emptyset^* = \varepsilon$
$\varepsilon^* = \varepsilon$
$L^+ = LL^* = L^*L$
$L^* = L^+ + \varepsilon$

Test for Reg Exp Algebraic Law:

To test whether $E = F$, where E, F are reg exp with variables $(L, M, N, \ldots)$, convert E, F to concrete reg exp C, D by replacing variables by symbols. If $L(C) = L(D)$, then $E = F$.

Ex. To show $(L + M)^* = (L^*M^*)^*$ replace L, M by a, b, to obtain $(a + b)^* = (a^*b^*)^*$.

Pumping Lemma: Let L be a regular language. Then there exists a constant n (depending on L) such that for all $w \in L$, $|w| \geq n$, we can break w into three parts $w = xyz$ such that:

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. For all $k \geq 0$, $xy^kz \in L$

Proof: Suppose L is regular. Then there exists a DFA A such that $L = L(A)$. Let n be the number of states of A. Consider any $w = a_1a_2\dots a_m$, $m \geq n$:

$$\underset{p_0}{\uparrow}\ \overbrace{a_1 \underset{p_1}{\uparrow} a_2 \underset{p_2}{\uparrow} a_3 \dots a_i}^{x}\ \underset{p_i}{\uparrow}\ \overbrace{a_{i+1} \dots a_j}^{y}\ \underset{p_j}{\uparrow}\ \overbrace{a_{j+1} \dots a_m}^{z}\ \underset{p_m}{\uparrow}$$

Ex. Show $L = \{0^n1^n | n \geq 0\}$ is *not* regular.

Suppose it is. By PL $\exists p$. Consider $s = 0^p1^p = xyz$. Since $|xy| \leq p$, $y \neq \varepsilon$, $y = 0^j$, $j > 0$. And $xy^2z = 0^{p+j}1^p \in L$, which is a contradiction.

Ex. Show $L = \{1^p |\ p \text{ is prime }\}$ is not regular.

Suppose it is. By PL $\exists n$. Consider some prime $p \geq n + 2$.

Let $1^p = xyz$, $|y| = m > 0$. So $|xz| = p - m$.

Consider $xy^{(p-m)}z$ which must be in L.

But
$|xy^{(p-m)}z| = |xz| + |y|(p-m) = (p-m) + m(p-m) = (p-m)(1+m)$

Now $1 + m > 1$ since $y \neq \varepsilon$, and $p - m > 1$ since $p > n + 2$ and $m = |y| \leq |xy| \leq n$. So the length of $xy^{(p-m)}z$ is not prime, and hence it cannot be in L — contradiction.

R is a *relation* on two sets A, B if $R \subseteq A \times B$.

e.g. $R = \{(m,n) |\ m - n \text{ is even } \} \subseteq \mathbb{Z} \times \mathbb{Z}$.
So $(3,5), (2,-4) \in R$, but $(-2,1) \notin R$.

R is an *equivalence relation* if it is

1. Reflexive: for all a, $(a,a) \in R$
2. Symmetric: for all a, b, $(a,b) \in R \Rightarrow (b,a) \in R$
3. Transitive: for all a, b, c, $(a,b) \in R$ and $(b,c) \in R$, implies that $(a,c) \in R$.

If R is an equivalence relation, and $(a,b) \in R$, then we write $a \equiv_R b$ or just $a \equiv b$.

Equivalence class: $[a] = \{x | x \equiv a\}$

Theorem: For any equivalence relation:

1. $a \in [a]$
2. $a \equiv b \iff [a] = [b]$
3. $a \not\equiv b$ then $[a] \cap [b] = \emptyset$
4. any two equivalence classes are either equal or disjoint.

Proof: 3. prove the contra-positive: suppose $[a] \cap [b] \neq \emptyset$, so there exists an $x \in [a] \cap [b]$.

By definition, $x \equiv a$ and $x \equiv b$.

By symmetry and transitivity, $a \equiv b$.

$L \subseteq \Sigma^*$; given $x, y \in \Sigma^*$ we say that they are *distinguishable* if $\exists z \in \Sigma^*$ such that exactly one of xz, yz is in L.

E.g., $L = \{w \in \{0,1\}^* |\ w$ has an even number of 1s $\}$, and $x = 00, y = 10$. Then x, y are distinguishable because letting $z = 1$, $xz = 001 \notin L$ but $yz = 101 \in L$.

Given L, let $\equiv_L$ be the relation: $x \equiv_L y$ iff x, y are *not* distinguishable. Then $\equiv_L$ is an equivalence relation.

Myhill-Nerode Theorem: L is regular $\iff$ $\equiv_L$ has *finitely many* equivalence classes.

Moreover, the number of states in the smallest DFA recognizing L is equal to the number of equivalence classes of $\equiv_L$.

Closure Properties of Regular Languages

Union: If L, M are regular, so is $L \cup M$.

Proof: $L = L(R)$ and $M = L(S)$, so $L \cup M = L(R + S)$.

Complementation: If L is regular, so is $L^c = \Sigma^* - L$.

Proof: $L = L(A)$, so $L^c = L(A')$, where A' is the DFA obtained from A as follows: $F_{A'} = Q - F_A$.

Intersection: If L, M are regular, so is $L \cap M$.

Proof: $L \cap M = \overline{\overline{L} \cup \overline{M}}$.

Reversal: If L is regular, so is $L^R = \{w^R | w \in L\}$, where $(w_1 w_2 \dots w_n)^R = w_n w_{n-1} \dots w_1$.

Proof: Given a reg exp E, define E^R by structural induction. The only trick is that $(E_1 E_2)^R = E_2^R E_1^R$.

Homomorphism: $h : \Sigma^* \longrightarrow \Sigma^*$, where
$h(w) = h(w_1 w_2 \dots w_n) = h(w_1)h(w_2)\dots h(w_n)$.

Ex. $h(0) = ab, h(1) = \varepsilon$, then $h(0011) = abab$.

$h(L) = \{h(w) | w \in L\}$

If L is regular, then so is $h(L)$.

Proof: Given a reg exp E, define $h(E)$.

Inverse Homomorphism: $h^{-1}(L) = \{w | h(w) \in L\}$.

Proof: Let A be the DFA for L; construct a DFA for $h^{-1}(L)$ as follows: $\delta(q, a) = \hat{\delta}_A(q, h(a))$.

Complexity of converting among representations

ε-NFA $\longrightarrow$ DFA is $O(n^3 2^n)$
$O(n^3)$ for computing the ε closures of all states – Warshall's algorithm, and 2^n states

DFA $\longrightarrow$ NFA is $O(n)$

DFA $\longrightarrow$ Reg Exp is $O(n^3 4^n)$
There are n^3 expressions $R_{ij}^{(k)}$, and at each stage the size quadruples (as we need four stage $(k-1)$ expressions to build one for stage k)

Reg Exp $\longrightarrow$ ε-NFA is $O(n)$
The trick here is to use an efficient parsing method for the reg exp; $O(n)$ methods exist

Decision Properties

- Is a language empty?
  Automaton representation: Compute the set of reachable states from q_0. If at least one accepting state is reachable, then it is not empty.
  What about reg exp representation?
- Is a string in a language?
  Translate any representation to a DFA, and run the string on the DFA.
- Are two languages actually the same language?
  Equivalence and minimization of Automata.

Equivalence and Minimization of Automata

Take a DFA, and find an *equivalent* one with a *minimal* number of states.

Two states are equivalent iff for all strings w,

$$\hat{\delta}(p, w) \textit{ is accepting } \iff \hat{\delta}(q, w) \textit{ is accepting}$$

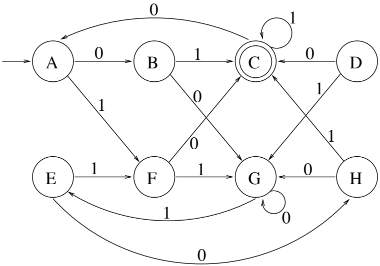
If two states are not equivalent, they are *distinguishable*.

Find pairs of distinguishable states: Basis Case: if p is accepting and q is not, then $\{p, q\}$ is a pair of distinguishable states.

Induction Step: if $r = \delta(p, a)$ and $s = \delta(q, a)$, where $a \in \Sigma$ and $\{r, s\}$ are distinguishable, then $\{p, q\}$ are distinguishable.

Table Filling Algorithm

A recursive algorithm for finding distinguishable pairs of states.



	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Distinguishable states are marked by "x"; the table is only filled below the diagonal (above is symmetric).

Theorem: If two states are not distinguished by the algorithm, then the two states are equivalent.

Proof: Use the Least Number Principle (LPN): any set of natural numbers has a least element.

Let $\{p, q\}$ be a distinguishable pair, for which the algorithm left the corresponding square empty, and furthermore, of all such "bad" pairs $\{p, q\}$ has a shortest distinguishing string w.

Let $w = a_1 a_2 \dots a_n$, $\hat{\delta}(p, w)$ is accepting & $\hat{\delta}(q, w)$ isn't.

$w \neq \varepsilon$, as then p, q would be found out in the Basis Case of the algorithm.

Let $r = \delta(p, a_1)$ and $s = \delta(q, a_1)$. Then, $\{r, s\}$ are distinguished by $w' = a_2 a_3 \dots a_n$, and since $|w'| < |w|$, they were found out by the algorithm.

But then $\{p, q\}$ would have been found in the next stage.

Equivalence of DFAs

Suppose D_1, D_2 are two DFAs. To see if they are equivalent, i.e., $L(D_1) = L(D_2)$, run the table-filling algorithm on their "union", and check if $q_0^{D_1}$ and $q_0^{D_2}$ are equivalent.

Complexity of the Table Filling Algorithm: there are $n(n-1)/2$ pairs of states. In one round we check all the pairs of states to check if their successor pairs have been found distinguishable; so a round takes $O(n^2)$ many steps. If in a round no "x" is added, the procedure ends, so there can be no more than $O(n^2)$ rounds, so the total running time is $O(n^4)$.

Minimization of DFAs

Note that the equivalence of states is an equivalence relation. We can use this fact to minimize DFAs.

For a given DFA, we run the Table Filling Algorithm, to find all the equivalent states, and hence all the equivalence classes. We call each equivalence class a *block*.

In our last example, the blocks would be:

$$\{E, A\}, \{H, B\}, \{C\}, \{F, D\}, \{G\}$$

The states within each block are equivalent, and the blocks are disjoint.

We now build a minimal DFA with states given by the blocks as follows: $\gamma(S, a) = T$, where $\delta(p, a) \in T$ for $p \in S$.

We must show that γ is well defined; suppose we choose a different $q \in S$. Is it still true that $\delta(q, a) \in T$?

Suppose not, i.e., $\delta(q, a) \in T'$, so $\delta(p, a) = t \in T$, and $\delta(q, a) = t' \in T'$. Since $T \neq T'$, $\{t, t'\}$ is a distinguishable pair. But then so is $\{p, q\}$, which contradicts that they are both in S.

Theorem: We obtain a minimal DFA from the procedure.

Proof: Consider a DFA A on which we run the above procedure to obtain M. Suppose that there exists an N such that $L(N) = L(M) = L(A)$, and N has fewer states than M.

Run the Table Filling Algorithm on M, N together (renaming the states, so they don't have states in common). Since $L(M) = L(N)$ their initial states are indistinguishable. Thus, each state in M is indistinguishable from at least one state in N. But then, two states of M are indistinguishable from the same state of $N \ldots$