CS310 : Automata Theory

Prof. Paritosh Pandya

Hopcroft, motwani : 1.2 - 1.4 (proof systems)

Introductory class

Automata Theory is known by :

1. Computability Theory
2. Theory of Computation
3. Formal language theory

{ Automata theory developed to complexity theory, though complexity will not be covered }

- Calculational method ≡ Algorithm, eg. gcd

1. model of System (Semantics of Prog. Lang)
2. Computer system of Implementation

- Turing machine : one of central ideas of this course

von-neumann architecture : ALU + Registers + control unit + Data bus + Memory (data & prog.)

[Stored program machine]

{ equivalence b/w iterative and recursive programs ? which is more powerful ? }

Recursive language

Primitive recursion: constant 0, $f(x)=x+1$

function composition: $f(g(x))$

Recursive function invocation: $f(x+1) = h(f(x))$

add(x, y)

$add(x,0) = x$

$add(x, S(y)) = S(add(x,y))$

mult(x, y)

$mult(x,1) = x$

$mult(x, y+1) = add(mult(x,y), x)$

Exp(x)

$exp(0) = 1$

$exp(y+1) = mult(2, exp(y))$

$2 \to S(S(0))$

Proof of $exp(n) = 2^n$ (induction on n)

Ackermann function

Because the semantics are not followed, 2nd parameter ↑

#ques: why doesn't fit into primitive recursion?

$A(0,y) = y+1$

$A(x+1, 0) = A(x,1)$

$A(x+1, y+1) = A(x, A(x+1, y))$

#ques: while model?

while programs are stronger than recursive ones, a stack of nums can be mapped to a single one using godel's numbering scheme (Gödel: general recursion conjectured to be as strong)

Models of computation

- Counter machines
- Godel's Mu Recursive Functions
- Turing Machines
- Lambda Calculus
- Post system
- Chomsky Unrestricted Grammars
- Programming Languages: Fortran, Pascal, C, Lisp, Java.

Blue will be covered, others maybe

Note : Termination of ackermann function can be shown by double induction

Study of Systems of Computations:

- (formal verification) What are the properties of a concrete algorithm given in a system of computation? What methods can be used to analyse such questions?
- (expressivity) What kind of calculations are possible in a system of computation?
  Can system A compute everything that is computable in system B?
- (computability) Is a problem solvable in given system of computation?

Explored first by logicians and mathematicians (Hilbert, Godel, Church, Turing, Post, Kleene, ...)

Turing : 1. Turing machine
2. Church - Turing machine
3. Univercal turing machines
(Turing - church hypothecis)
4. Undecidability of halting problem
(absolute limits to computation)

Hypothecis : Any model of computation can be mimicked by a turing machine

---

Computations which can be carried out with only a bounded amount of memory.

Theorem

Every Sequential Circuit can be modelled as an equivalent Finite State Automaton and vice versa.

Given sequential circuits A and B, do they have the same behaviour?
Answered by checking if $Aut(A) \equiv Aut(B)$.

Applications of Finite Automata

- Design of Digital Circuits
- Communication protocols, computer architecture
- Discrete control systems
- Analysis of correct behaviour of above systems (model checking)
- Regular expression pattern matching (most programming languages)
- Construction of lexical analyzers in compiler design
- Natural language processing (tokenization)

#ques : recursive progs are as hard as number theory (fermat's conjecture)

#ques: Is there a semantic for models of computation (i.e. what is a model of computation over which turing machine is most general)

#quel: Is this same as symbolic algo? (this = recursion)

Finite State Automata

Kozen: lectures 2 & 3
Hopcroft, Motwani & Ullmann: chap 1 & 2

Decision Problem (yes or no questions)

e.g. Given a directed graph, constant k,
Is there a path of length atmost k?

A: set of candidates, instances (e.g. $\mathbb{N}$)
B: subset of A for which answer is yes (e.g. primes)
given $x \in A$ to determine if $x \in B$

Automaton for (A, B)
A machine which can solve any given instance of a decision problem (A, B)

Test for eulerian (return to same vertex) # To do: give a proof
G is eulerian iff $\forall_v$ d(v) is even and G is connected

Standard formulation of instances (finite set)
Encode each input as a string of alphabet (Σ)
Unary encoding of n: repeat "a" n times ($\Sigma = \{a\}$)
0 is empty string
$= a^n$ (notation)

Graph inputs e.g. adjacency list
(flatten into string encoding)

Instance of a problem: finite word

Todo : insert grammar here

Notation : $x, y, z, \ldots$ will denote words

$x \cdot y$ and xy is same thing

concatenation with alphabet forms monoid

$(S, \cdot)$ is a monoid if $\cdot$ is associative and there is an identity, i.e. semigroup with identity.

if every element has an inverse then group

$x^{n+1} = (x^n)\, x$ { recursive definition }

Structural induction is v. important

Defining length $|x| : \Sigma^* \to \mathbb{N}$

$|\epsilon| = 0$

$|ua| = |u| + 1$

Language over $\Sigma = L(\Sigma) \subseteq \Sigma^*$

e.g. $\Sigma = \{a\}$, $A = \{a^p \mid p \text{ is prime}\}$

$\Sigma = \{0,1\}$, $x \in \Sigma^*$, $\#0(x)$ = count of zeros,

$\#1(x)$ = count of ones. $A = \{x \in \Sigma^* \mid \#0(x) = \#1(x)\}$

(# can be defined inductively again)

$A = \{x \in \Sigma^* \mid \exists i \in \mathbb{N}$ s.t. x is binary repr of $2^i\}$

$\Sigma = \{a, \ldots, z\}$. palindromes are language

$\Sigma =$ ASCII. valid C programs are language

Let $\Sigma = \{0,1\}$. $A = \{0^n 1^n \mid n \geq 0\}$

#Notation: $A = 0^n 1^n$ is equivalent notation

Operations on languages: 1. Usual algebra over sets

#To do: Cozec chapter 2 $(\cup, \cap, \sim)$ (same alphabet for both languages)

2. Concatenation (forms a monoid over langs as well)

$$L_1 \cdot L_2 = \{a \cdot b \mid a \in L_1 \text{ & } b \in L_2\}$$

(languages can be infinite as Σ^* is infinite)

e.g. $\{a^p \mid p \text{ is prime}\}$

Props: $A \cdot (B \cup C) = A \cdot B \cup A \cdot C$

Language recognition problem : $x \in \Sigma^*$, does $x \in A$
(A is a language)

All computational decision problems can be encoded as language recognition problems

e.g. $A_1 = \{x \in \{a, b\}^* \mid \#a(x) = \#b(x)\}$

1 counter program (inc, dec)

ques : Why is induction an axiom over structures, and does it depend on the property of the structure ?

1. Unique homomorphic extension theorem
A helper f^n which applied to satisfy structural def^n is unique

↓ must satisfy an underlying inductive property

2. Free generation
elements of lang when measured as # f^ns applied (e.g. length = #power), so, number of f^ns should be unambiguous.

Short answer : Yes, it depends on the structure,
(works for context-free grammars)

#ques: ∃ problem solvable 2 counters but not 1 counter

#ques: Why can any computational decision problem be modelled as a language detection over words

Finite State Automata

(5/8/23)

To do : paste list of topics

DFA

e.g. light with toggle switch.

Actions = Σ = {press} (alphabet)

set of states

$Q = \{off, on\}$

$q_0 = off$

(transition diagram representation)

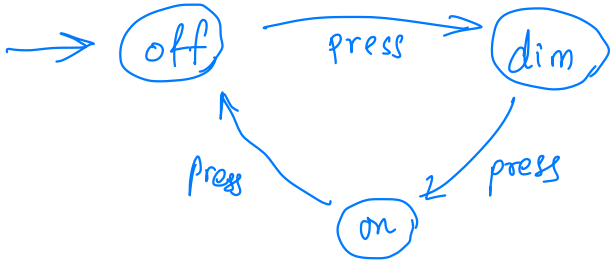
Transition function ($\delta : Q \times \Sigma \to Q$)

$\delta(off, press) = on$

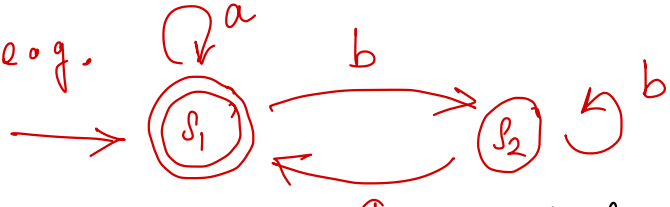
$\delta(on, press) = off$

off $\xrightarrow[press]{}$ on $\xrightarrow[press]{}$ off $\xrightarrow[press]{}$ on

e.g. dimmable light



e.g.



For a given word, the run is unique

Accepting run : define set of final states. Final states $F \subseteq Q$. Run is accepting if end state in F (accepts a language)

Language : $\mathcal{L}$ or words of form ua, $u \in \Sigma^*$

Formal definition of automaton

$Q = \{S_1, S_2\}$

$\Sigma = \{a, b\}$

$q_0 = S_1$

$F = \{S_1\}$

$\delta = \{(S_1, a, S_1), (S_1, b, S_2), (S_2, b, S_2), (S_2, a, S_1)\}$

(δ is a total function : defined on entire $Q \times \Sigma$)

↳ used in theorems about DFA

1. 5-tuple method $(Q, \Sigma, q_0, F, \delta)$
2. Transition diagram
3. Transition table

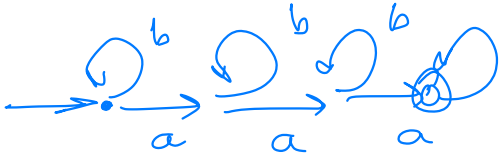
e.g.

	a	b
0	1	0
1	2	1
2	3	2
3F	3	3

} alphabet

states

final state has F next to it



$L(A) = \{x \in \Sigma^* \mid n_a(x) \geq 3\}$

$q \xrightarrow{a} q'$ denotes $\delta(q, a) = q'$

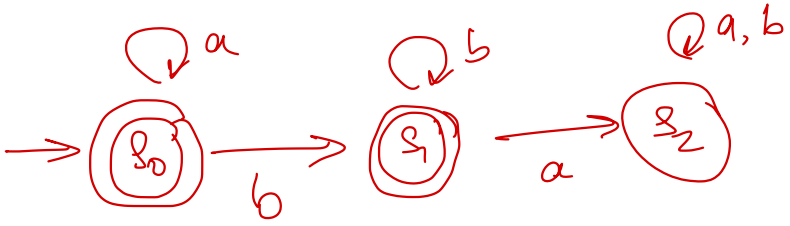
- A run of DFA A on $x = a_0, a_1, \ldots, a_{n-1}$ is a sequence of states $q_0, q_1, \ldots, q_n$ s.t. $q_i \xrightarrow{a_i} q_{i+1}$ for $0 \leq i \leq n$
- For a given word x, the DFA has a unique run
- A run is accepting if last state $q_n \in F$

syntax = just defn. semantics = run, accepting word, etc.
(doesn't mean anything)

$L(A) = \{u \in \Sigma^* \mid$ The run of A on u is accepting $\}$

Language is regular if accepted by a DFA.

ques: Does regularity of a problem depend on encoding?

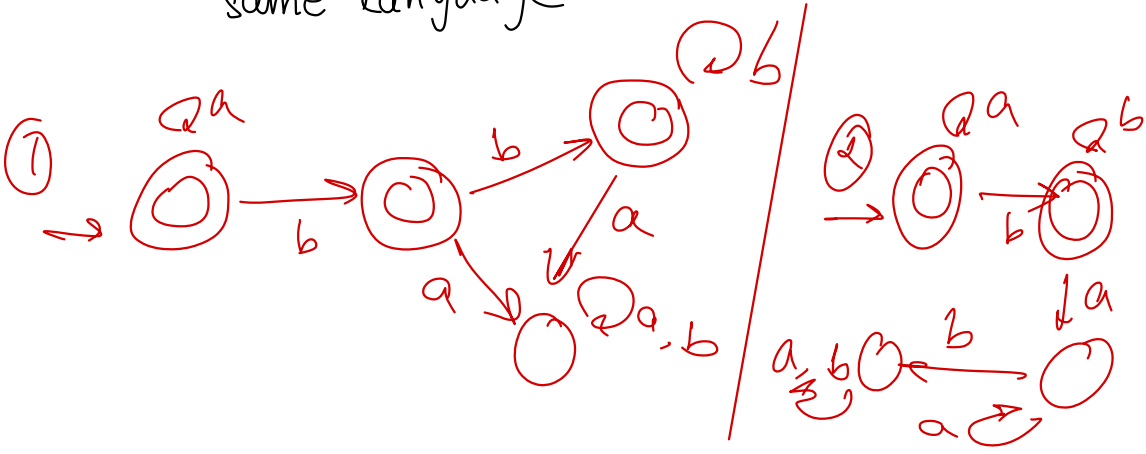


$s_0 = (\#b = 0)$

$s_1 = (\#b > 0)$

$s_2 = (b$ preceding $a)$

ques: Can we have non-isomorphic automata with same number of states, encoding the same language

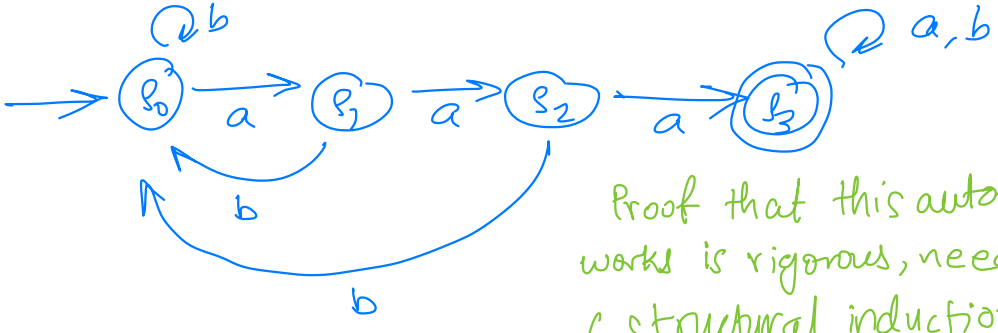


$x \leq_p y \triangleq$ (defined as) $\exists_z \; x \cdot z = y$ (prefix)

$x \leq_s y \triangleq \exists_{u,v} \; uxv = y$ (subword)

subword 'aaa'

(matching first occurence)



Proof that this automata works is rigorous, needs math (structural induction)

Lecture 3 (Generation of natural numbers) (7/8/23)

Basis elements : 0

Constructor $S : n \mapsto n+1$ (over $\mathbb{N}$)

{inductive/recursive defined}

set should be "freely generated" (unique way of gen from basis)

Unique homomorphic extension theorem (UHET)

X_+ is a free set generated by X (basis) and f (functions), for each fn $h : X \to B$ $\exists$ single $\hat{h} : X_+ \to B$ s.t.

$\forall_{x \in X} \hat{h}(x) = h(x)$

Over words Universe $= \Sigma^*$

Basis $= \varepsilon$

(e.g. $\Sigma = \{a, b, c\}$) $\alpha_a : \Sigma^* \to \Sigma^*$ | appends a, b, c (constructor) at the end of string

α_b :

α_c :

$\forall_{x \in \Sigma^*} \left(P(x) \Rightarrow \forall_{a \in \Sigma} P(\alpha_a(x)) \right)$ iff $\forall_{x \in \Sigma^*} P(x)$

(induction step)

len(x)

Base step : $len(\varepsilon) = 0$

$len(\alpha_a(x)) = len(xa) = len(x) + 1$

By UHET, len is unique f^n

$U \in \Sigma^*$, $\Sigma = \{a, b, c\}$ (even length palindromes) } PALIN

Basis $= \varepsilon$

Constructor $\forall_{x \in \Sigma}$ $\alpha_a : x \mapsto axa$

Claim : $\forall x \in PALIN$, $len(x)$ is even

Basis : $len(\varepsilon) = 0$

Induction : $len(\alpha_a(x)) = len(axa) = len(x) + 2$

$len(x)$ is even

Hence, $len(\alpha_a(x))$ is even ∎

Exercise : Numbers in unary

Powers of 2 = { a, aa, aaaa, ... }

Numbers in binary

= {1, 10, 100, ...}

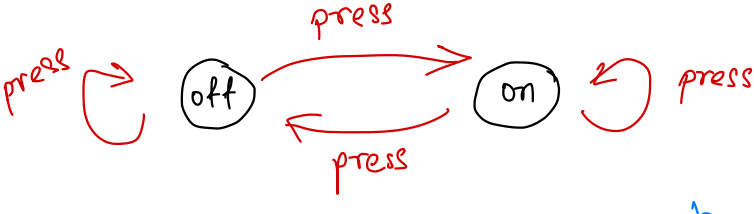
Binary num (with leading zeros) divisible by 3 (trivial, done in Kozen)

mod 0, mod 1, mod 2 states for binary

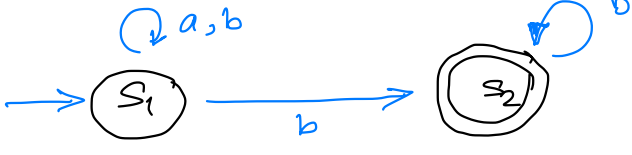
lecture 4

(08/08/23)

NFA



zero or more edges from state corresponding to alphabet



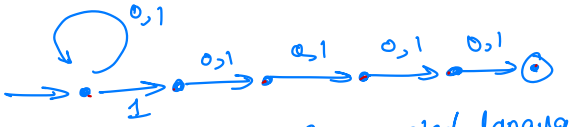
Is NFA a machine ?

(NFA - DFA construction)

$L = \{x \mid \exists u, v, x = u1v, v \in \Sigma^4\}$

$= \Sigma^* \{1\} . \Sigma^4$

Imp : Explicit construction for 2^n states



(5th last letter is a 1) = accepted language

Claim : There is no DFA M with $|M| < 2^5$ and $L(M) = L(N)$

Proof : Assume to contrary, such an M exists.

for 2^5 paths, atleast two end up in the same state, then, extend them so that one should be accepted, one rejected.

Construct the paths as $(0,1)^5$. Let two paths end up in same state, differ in first position i,

e.g. 0 1 0 1 1 0 0 — now extend with 4-i zeros.

0 1 1 1 0 0 0

They must end up in the same state but one run is accepting and the other isn't ▨

Lecture 5
(14/08/2023)

Quiz : 31/08/23 (Monday), 28/08 : Doubt Solving

Syllabus : Upto minimization of DFA

$L_1 \cap L_2$: $(Q_1 \times Q_2, \ldots)$ Final states if $q_1 \in F_1, q_2 \in F_2$

$L_1 \cup L_2$: $(Q_1 \times Q_2, \ldots)$ if $q_1 \in F_1$ or $q_2 \in F_2$

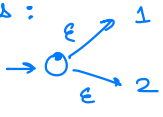
$\sim L_1$: final states are $Q \setminus F$

Complexity of closure operations :

	DFA	NFA
• Complement	m	2^m (determinising then complement, acceptance cond" issue)
• Intersect	$m \times n$	$m \times n$
• Union	$m \times n$	$m+n$ (place wise)

(#quel : what is ε idea ?)

ans :

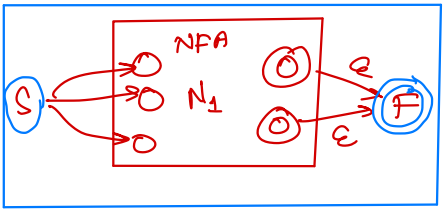


+ m+n NFA, so worst case 2^{m+n}

Normalised ε- NFA

1. Single start & single final state
2. No incoming transitions to start
3. No outgoing transitions from final state.

Given ε-NFA, we can construct normalised ε-NFA



N_2

Proof : Any word accepted by N_1 is accepted by N_2 and vice versa. Easy to show.

Theorem

Given normalized ϵ-NFA $N_1 = (Q_1, \Sigma, \Delta_1, s_1, f_1)$ and $N_2 = (Q_2, \Sigma, \Delta_1, s_2, f_2)$ we can construct normalized ϵ-NFA $N_3 = (Q_3, \Sigma, \Delta_3, s_3, f_3)$ s.t. $L(N_3) = L(N_1) \cup L(N_2)$.

(5-tuple construction for union of NFAS normalised ε- NFA)

- $Q_3 = Q_1 \cup Q_2 \cup \{s_3, f_3\}$ fresh states s_3, f_3.
- $\Delta_3(s_3, \epsilon) = \{s_1, s_2\}$, and $\forall a \in \Sigma.\ \Delta_3(s_3, a) = \emptyset$.
- $\forall s \in Q_1.\ \Delta_3(s, a) = \Delta_1(s, a)$ and $\forall s \in Q_2.\ \Delta_3(s, a) = \Delta_2(s, a)$.
- $\forall s \in Q_1 - \{f_1\}.\ \Delta_3(s, \epsilon) = \Delta_1(s, \epsilon)$, and $\forall s \in Q_2 - \{f_2\}.\ \Delta_3(s, \epsilon) = \Delta_2(s, \epsilon)$.
- $\Delta_3(f_1, \epsilon) = \Delta_1(f_1, \epsilon) \cup \{f_3\}$ and $\Delta_3(f_2, \epsilon) = \Delta_2(f_2, \epsilon) \cup \{f_3\}$

Catenation

$A \cdot B = \{x \cdot y \mid x \in A \text{ and } y \in B\}$

normalised ε NFAs $N_1 \to N_2$, then add ε-transition from F of N_1 to S of N_2.

#ques: if no ε-normalisation? → no problem (ε-normalisation is optional)

Kleene closure

$NP = \{A, An, the\} \cup \{\varepsilon\} \cdot (Adjective)^* \cdot (Noun)$

optional — optional, 1 or more

the black cat
the nasty black cat

Let $A \subseteq \Sigma^*$, define $A^n = \{a_1 a_2 \dots a_n \mid a_i \in A\}$

$A^{n+1} = A \cdot A^n$ (inductive def^n).

$\{ab, aab\}^0 = \{\varepsilon\}$

$\{ab, aab\}^1 = \{ab, aab\}$

$\{ab, aab\}^2 = \{abaab, aabaab, abaab, aaabaab\}$.

$A^* = \bigcup_{c \geq 0} A^c = A^0 \cup A^1 \cup A^2 \cup \dots$ (kleene closure)

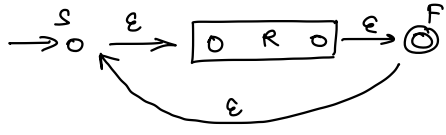
e.g. $A = \{ab\}$, then $A^* = \{(ab)^n \mid n \geq 0\}$

$L = \{x \mid \text{fifth last letter is 1}\}$

$= \Sigma^* \cdot \{1\} \cdot \Sigma^4$

Kleene closure of regular language is regular.

normalised ε-NFA $N_1 \to N_2$



Issue:

ε doesn't work as start state would have to be made accepting! ($b\varepsilon = b$ is accepting)

$(a^*b)^*$ ← ε is accepted.

issue is prev. construction accepts $b\varepsilon a = ba$!

* letter a maps to some string in b

Homomorphism:

$h: \Sigma^* \to \Gamma^*$ s.t.

$\forall u, v \in \Sigma^* \quad h(u \cdot v) = h(u) \cdot h(v)$

Consequences.

1. $h(\varepsilon) = \varepsilon$ — if $h(\varepsilon) = x$ then $h(\varepsilon \cdot y) = h(y) = x \cdot h(y)\ \forall x$. hence $x = \varepsilon$
2. Giving $h(a)$ for $a \in \Sigma$ uniquely determines h

If $h: \Sigma^* \to \Gamma^*$ is a homomorphism and $B \subseteq \Gamma^*$ is regular then $h^{-1}(B) = \{x \in \Sigma^* \mid h(x) \in B\}$ is also regular.

Proof: Given a DFA $BB = (Q, \Gamma, \delta, q_0, F)$ for B construct DFA AA for $h^{-1}(B)$ as

$AA := \{Q, \Sigma, \delta', q_0, F\}$ with $\delta'(q,a) = \hat{\delta}(q, h(a))$

Proof that this is correct follows from showing that accepted words are the the same.

If $h: \Sigma^* \to \Gamma^*$ and $A \subseteq \Sigma^*$ is regular then $h(A) = \{h(x) \mid x \in A\}$ is also regular.

Pf: replace string with chain of useless states!

Refer: Kozen for proof of homomorphism theorems.

Example: Let $x \in \{0,1\}^*$

$H_3(x) = \{y \mid$ hamming distance between $x, y \leq 3, |x| = |y|\}$

$H_3(A)$ = set of all pointwise dist 3 from elements of A.

- If A is regular, $H_3(A)$ is regular as
  1. Copy states of A 4 times
  2. Have directed transitions for complement bits to copied DFAs.

Alternative: $\Sigma = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$

$top\left(\begin{bmatrix}x\\y\end{bmatrix}\right) = x$. $bottom\left(\begin{bmatrix}x\\y\end{bmatrix}\right) = y$.

Say $x = \begin{matrix}0&1&0&1&0\\0&0&0&1&1\end{matrix}$ $top(x) = 01010$

$D_3 \subseteq \Sigma^* = \{x \in \Sigma^* \mid d_H(top(x), bottom(x)) \leq 3\}$

A_3 is automaton for D_3 (easy to make)

Given $A \subseteq \{0,1\}^*$, $H_3(A) = bottom(top^{-1}(A) \cap D_3)$

(adds bottom track; 2nd track differs at atmost 3 places; take only bottom track.)

So, H_3 can be built as top, bottom are homomorphisms, over words.

$\Sigma = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$

$\Gamma = \{0,1\}$

$h: \Sigma^* \to \Gamma^*$

(H_3 is over Γ^*)

$A/B = \{x \mid \exists y \in B \; xy \in A\}$

Prefix(A) ← all states from which F reachable are accepting

Suffix(A) ← mult. start state. reachable from s.

Infix(A) ← intx ideas.

#ques → First Half$(A) = \{x \mid \exists y \; |x| = |y|$ and $xy \in A\}$

1. Collapse A using Σ transitions (call A')
2. Simultaneously run $A, (A')^R$ and add accepting iff both reach same state.

Regular Expressions : refer slides

Kozen : ref homomorphism proof

Non-deterministic (angel)
n-states
$\Rightarrow \log n$ bits (for bits)
(NL-algorithms)

Decision problems :

1) membership $x \in L(N)$? simulate N on x

2) Non-emptiness : check s - F reachability (DFS / BFS)

regular : check if reg-doesn't have $\emptyset$. even if it does

has ε (reg) ← exercise #

3) Language equivalent : given NFA N_1, N_2, determine if $L(N_1) = L(N_2)$

we can reduce this to checking $L(N_1) \subseteq L(N_2) \wedge L(N_2) \subseteq L(N_1)$

$A \subseteq B$ iff $(A \cap (\sim B)) = \emptyset$

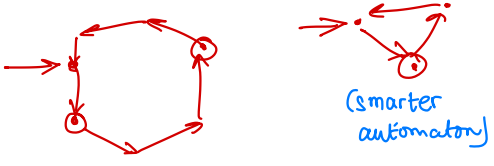
check if Product $(N_1, \text{compl}(N_2)) = \emptyset$

$|N| = n_1 \cdot 2^{n_2}$

time for emptiness $= O((n_1 \cdot 2^{n_2})^2)$

#edges.

Multiple DFAs for same language



ε-NFA to DFA conversion often gives rise to large automaton which may not be "optimal" in size.

Q. How to find optimal DFA for a DFA?

Q. Is it unique? (unique = isomorphic to other optima)

Essentially : p, q states are distinguishable if $\exists x$ s.t.

ps is accep but qs rejects or vice versa.

$p \approx q \triangleq \forall x \in \Sigma^*. \ (\hat{\delta}(p,x) \in F \iff \hat{\delta}(q,x) \in F)$

↑ reflexive, transitive, symm (equivalence rel^n)

$p \approx p$ (trivially)

$p \approx q \Rightarrow q \approx p$ (def^n is symm)

$p \approx q, q \approx r \Rightarrow p \approx r$. (easy to show)

(S, R) Equivalence rel^n if reflexive, transitive, symmetric.

↑ set ↑ $R \subseteq S \times S$.

e.g. $(\mathbb{N}, R)$, xRy iff $x \bmod 3 = y \bmod 3$

Equivalence partitioning $\{0, 3, 6, 9, \ldots\}$

$\{1, 4, 7, \ldots\}$

$\{2, 5, 8, \ldots\}$

Note: Eq. class has either all final or all non-final states! (ε distinguish)

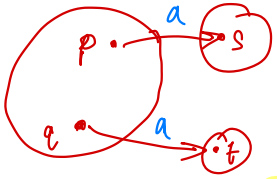
$(X/\sim \triangleq \{[x] \mid x \in X\}$ (Quotient set)

Quotient Automata

↓

Collapse eq. class into 1 state $M/\hat{\sim}$ quotient automata

$[i]$ ↖ equivalence class of i.



not possible

If so, $\exists\, w$ s.t. $\hat{\delta}(s, w) \in F$, $\hat{\delta}(t, w) \notin F$

then aw is dist. word for p, q.

Hence, now transitions can be defined!

TBD: Quotienting collapses once, but is it the "best"?

↳ yes! PHP. If $\exists$ $n-1$ state (i^{th} eq. class).

Let w_i s.t. $\hat{\delta}(s, w_i) = i_i$ $i = 1$ to n.

then $\hat{\delta}(s', w_i) = \hat{\delta}(s', w_j)$ for some $i \neq j$

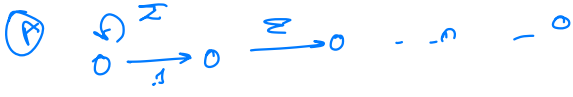
$\Rightarrow \hat{\delta}(s', w_i y) = \hat{\delta}(s', w_j y)$

But, s_i, s_j have y as disting. word hence

$\hat{\delta}(s', w_i y) \neq \hat{\delta}(s', w_j y)$

Hence, contradiction ∎

$\Sigma = \{0,1\}$. $L = \Sigma^* 1 \Sigma^{n-1}$.



(n+1 states)

Imp Note: reverse of this is a DFA! ($A^R = B$) So, reverse of B ∄ exponential blowup.

Claim: There is no DFA with $< 2^n$ states accepting L.

$\exists x, y \in \Sigma^n$ s.t. $x \neq y$ and $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$ (PHP)

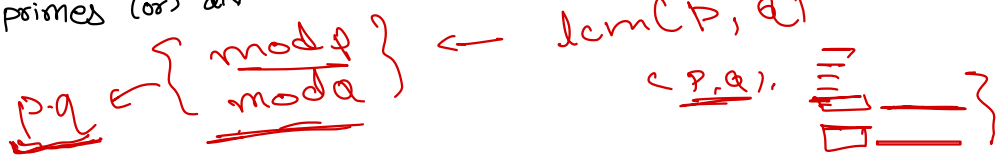
WLOG $x = u1z$, $y = v0z$ $\quad w = 0^{n-|z|-1}$

$|0zw| = |1zw| = n$.

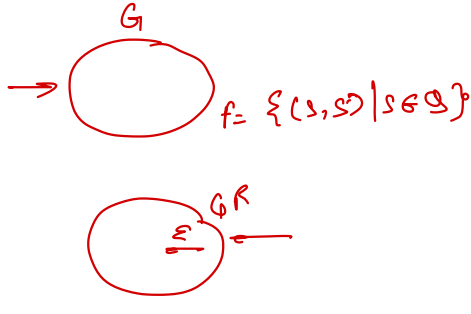
$u1zw \in L$, $v0zw \notin L$

Q. first n primes (or) div no succint DFA. (atleast lcm(p,q,...) states)

soln:



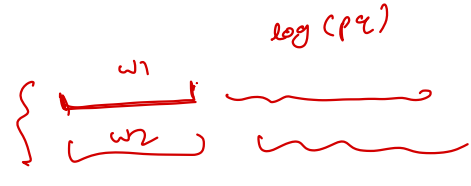
First half



Second half

pq nos. with diff (mod p, mod q)

if two in same state, WLOG mod p is different



$2^{\log pq}(w_1) = (p_1, q_1)$

$2^{\log pq}(w_2) = (p_2, q_2)$

$p_1 \neq p_2$ (WLOG)

1) say $q_1 \neq q_2$

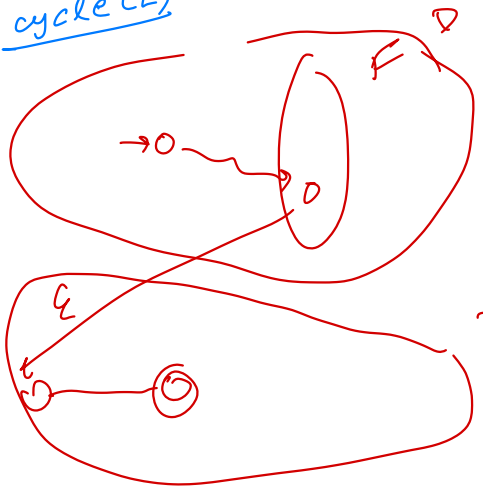
then find k s.t.

$k = (-p_1, -q_1)$

2) say $q_1 = q_2$

then find $k = (-p_1, -q_1 - 1)$

cycle(L)



$L \rightarrow$ regular

$cycle(L) = \{vu \mid uv \in L\}$

n copies.

Second half

$Q' = Q \times 2^Q \cup \{s, f\}$

$s \xrightarrow{\varepsilon} \{q, \{q\}\} \quad \forall q \in Q$

$\{q, A\} \xrightarrow{a} \{q', A'\}$

if $q \xrightarrow{a} q'$

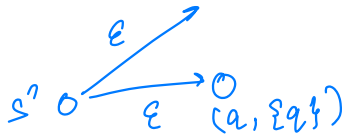
$A' = \{q'' \mid \exists a \in \Sigma \; q' \xrightarrow{a} q'' \in A\}$

if $q \in F$

$S \in A$

↑

original



$$\delta'(\{q, A\}, a) = (\delta(q, a), \{q' \mid \exists a' \in \Sigma\ \delta(q', a') \in A\}$$

Note: Any rational part x/y can be extracted by repeated cuts of $y/y+1$ fraction.

Note: Similarly any fraction can be constructed using first half, 2nd half ideas by constructing new edges = paths of length k to have first $k/k+1$, last $k/k+1$, first $1/k+1$, last $1/k+1$.