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)

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

Instance of a problem: finite word
To do: 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 $|x| : \Sigma^* \to \mathbb{N}$
$|\varepsilon| = 0$
$|ua| = |u| + 1$

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

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 (must satisfy an underlying inductive property)
A (helper f^n which applied to satisfy structural def^n is unique
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, (useful for context-free grammars)

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

	a	b
$\to q_0$	1	0
1	2	1
2	3	2
3F	3	3

e.g. — states — final state has ⊂ next to it — } alphabet

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

$q \xrightarrow{a} q'$ 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)

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

Basis elements : 0

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

{inductive/recursive defined}

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

Unique homomorphic extension theorem (UHET)

X_+ is a free set generated by X (basis) and f (functions), for each f^n $h : X \to B$ & 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. $\Sigma = \{a, b, c\}$) $\alpha_a : \Sigma^* \to \Sigma^*$, α_b :, α_c : — appends a, b, c (constructor) at the end of string

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

(Induction step)

len(x)

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

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

By UHET, len is unique fn

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

Constructor Basis $= \varepsilon$

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

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

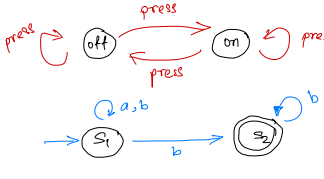
= {1, 10, 100, ---}

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

mod 0, mod 1, mod 2 states for binary

Lecture 4 (08/08/23)

NFA



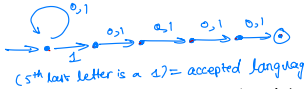
Is NFA a machine ?

(NFA-DFA construction)

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

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

Imp : Explicit construction for 2^n states



(5^{th} 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 co 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
0 1 1 1 0 0 0

now extend with 4−i zeros.

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 $q_2 \in F_2$

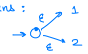
$\sim L_1$: final states are $Q \setminus F$

Complexity of closure operations:

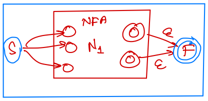
	DFA	NFA
Complement	m	2^m (determinising then complement, acceptance cond^n issue)
Intersect	$m \times n$	$m \times n$
Union	$m \times n$	$m+n$ (place $\|^{el}$)

(#ques: what is e idea?)

ans: $m+n$ NFA, to worst case 2^{m+n}



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



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

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 $N_2 = (Q_2, \Sigma, \Delta_2, s_2, f_2)$ 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, 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} ∪ {ε} . (Adjective)* . (Noun)
(optional) (optional, 1 or more)

the black cat
the nasty black cat

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

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

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

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

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

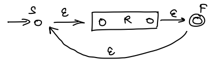
$A^* = \bigcup_{c \geq 0} A^c = A^0 \cup A^1 \cup A^2 \cup \ldots$ (kleene closure)

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

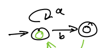
$L = \{x \mid$ fifth last letter is 1$\}$
$= \Sigma^* . \{1\} . \Sigma^4$

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 $b\varepsilon a = ba$!

Homomorphism:

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

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

* letter a maps to some string in b

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 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: $\Sigma = \{[^0_0], [^0_1], [^1_0], [^1_1]\}$

$top([^x_y]) = x$. $bottom([^x_y]) = y$.

say $x = {}^{01010}_{00011}$ $top(x) = 01010$

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

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.

$\Sigma = \{[^0_0], [^0_1], [^1_0], [^1_1]\}$

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

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

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) = $\{x \mid \exists y \; |x| = |y| \text{ and } xy \in A\}$

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
⇒ 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 ∅. 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 $L(N_1) \subseteq L(N_2)$ ∧ $L(N_2) \subseteq L(N_1)$

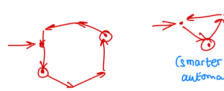
$A \subseteq B$ iff $(A \cap (\sim B)) = \phi$

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

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

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

Multiple DFAs for same language



(smarter automaton)

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 ∃ suffix s 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 $\{0, 3, 6, 9, \ldots\}$, $\{1, 4, 9, \ldots\}$, $\{2, 5, 8, \ldots\}$

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

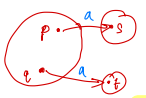
$(X/_{\sim} \triangleq \{[x] \mid x \in X\})$ (quotient set)

$[i]$ ↖ equivalence class of i.

Quotient Automata

↓

Collapse eq. class into 1 state $M/_{\sim}$ quotient automaton



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 $i \neq j$

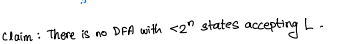
$\Rightarrow \hat{\delta}(s', w_i y) = \hat{\delta}(s', w_j y)$

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 ∎

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



Std Note: reverse of this is a DFA! ($\Phi^R = \beta$) So, reverse of β ∃ 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)

w.l.o.g. $x = u1z$, $y = v0z$, $w = 0^{n-|z|-1}$

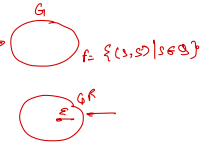
$(0^n w) = (1^n w) = n$.

$uw \in L$, $vw \notin L$

Q. first n primes (or) div no succint DFA. (atleast $\text{lcm}(p, q, \ldots)$ states)

Soln:

First half

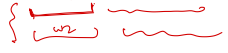


Second half

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

if two in same state, w.l.o.g. mod p is different

$\log(pq)$



$2^{\log pq}(w1) = (p_1, q_1)$

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

$p_1 \neq p_2$ (w.l.o.g)

1) say $q_1 \neq q_2$

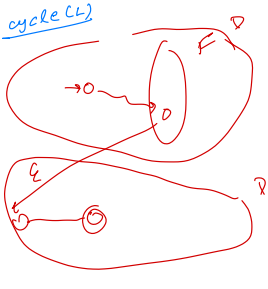
then find k s.t.

$k = (-p_1, -q_1)$

2) say $q_1 = q_2$

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

cycle(L)



$L \rightarrow$ regular

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

Reqd } n copies.

Second half

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

$s \xrightarrow{\varepsilon} \{q, \{q\}\}$ $\forall q \in P$

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

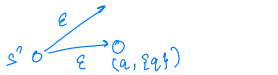
if $q \xrightarrow{a} q'$

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

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: 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 $1/k+1$.

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

- ϕ, No word matches. $L(\phi) = \emptyset$.
- a matches the single letter $a \in \Sigma$. $L(a) = \{a\}$
- ϵ Only empty word matches. $L(\epsilon) = \{\epsilon\}$.
- $\#$ Matches any letter from Σ. $L(\#) = \Sigma$
- $@$ Matches any word from Σ. $L(@) = \Sigma^*$

We use operations like catenation, intersection on languages to get more complex patterns.

- $(\alpha + \beta)$. Union. $L((\alpha + \beta)) = L(\alpha) \cup L(\beta)$.
- $(\alpha \cdot \beta)$. Catenation. $L((\alpha \cdot \beta)) = L(\alpha) \cdot L(\beta)$.
  We will typically write $\alpha \cdot \beta$ as $\alpha\beta$.
- (α^*). Kleene Closure.
  $L((\alpha^*)) = L(\alpha)^* = \bigcup_{n \geq 0} L(\alpha)^n$.
- Example: Compound Pattern $(((a^*) \cdot b) + (b \cdot (a^*)))$. Its language is what?
- $(\alpha \cap \beta)$. Intersection. $L((\alpha \cap \beta)) = L(\alpha) \cap L(\beta)$.
- $(\sim \alpha)$. Complement.
  $L((\sim \alpha)) = \sim L(\alpha) = \Sigma^* - L(\alpha)$.

Redundant Operators

- Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$. Then $\# \equiv (a_1 + a_2 + \ldots + a_n)$.
- $@ \equiv \#^*$.
- $\alpha \cap \beta \equiv \sim(\sim \alpha \cup \sim \beta)$.
- $\sim$ is also redundant (difficult proof).

Regular Expressions

Pattern expressins without $@, \#, \cap, \sim$. Abstract syntax: let α, β range over *RegExp*.

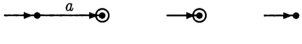
$$\alpha ::= a \mid \phi \mid \epsilon \mid \alpha \cdot \beta \mid \alpha + \beta \mid \alpha^*$$

Theorem

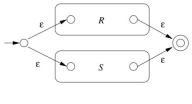
For every regular expression α we can construct a normalized ϵ-NFA $A(\alpha)$ such that $L(\alpha) = L(A(\alpha))$.

Proof (By stuctural induction). We construct the automaton bottom up.

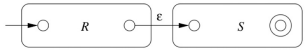
- Normalized ϵ-NFA for a, ϵ, ϕ respectively.



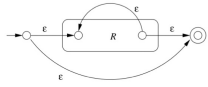
- For $\alpha + \beta$, inductively assume that we have normalized ϵ-NFA $A(\alpha)$ and $A(\beta)$. Construct union automaton recognizing $L(A(\alpha)) \cup L(A(\beta))$ as before.



- For $\alpha \cdot \beta$, inductively assume that we have normalized ϵ-NFA $A(\alpha)$ and $A(\beta)$. Construct catenation automaton recognizing $L(A(\alpha)) \cdot L(A(\beta))$ as before.



- For α^*, inductively assume that we have normalized ϵ-NFA $A(\alpha)$. Construct Kleene closure automaton recognizing $L(A(\alpha))^*$ as before.



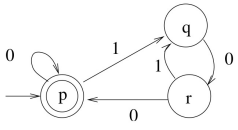
- size of regular expression α counts number of atomic expressions and number of composition operators in α. This is denoted $|\alpha|$.
  E.g. $|(a+b)^*aaa| = 10$ (+, *, 3 ·, 5 vars)
- Claim The size $|A(\alpha)| = O(|\alpha|)$. How?
  Prove by induction on the structure of regular expression α that $|A(\alpha)| \leq 2 * |\alpha|$.
  - (Base step) For α equals $a \mid \phi \mid \epsilon$ check that $|A(\alpha)| \leq 2$.
  - (Induction step 1) Let $\alpha = (\beta + \gamma)$.
    $|A(\alpha)| = 2 + |A(\beta)| + |A(\gamma)|$ (const. $A(\alpha)$),
    $\leq 2 + 2 * |\beta| + 2 * |\gamma|$, (ind. hyp.)
    $= 2(|\alpha|)$. (defn of $|\alpha|$).
  - Similar induction steps for $\beta \cdot \gamma$ and β^*

Theorem

For every NFA $A = (Q, \Sigma, \delta, I, F)$, we can construct a langauge equivalent regular expression $reg(A)$.

Construction Define regular expression $\alpha^X_{p,q}$ for $X \subseteq Q$ and $p, q \in Q$. This denotes set of words w such that A has a run on w from p to q where all intermediate states are from X. (via some sort of DP)

Example:



$\alpha^q_{p,r} = 1 \cdot 0$ $\alpha^{p,q}_{p,r} = 0^* \cdot 1 \cdot 0$

Aim: To compute $\alpha^{p,q,r}_{p,p}$

$\alpha^X_{p,r}$ can be recursively defined using the following rules:

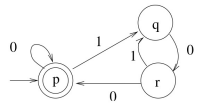
- $\alpha^{\emptyset}_{p,r} = a_1 + \ldots + a_k$ where $r \in \delta(p, a_i)$ (Δ our NFA) provided $p \neq r$
- $\alpha^{\emptyset}_{p,p} = a_1 + \ldots + a_k + \epsilon$ where $p \in \delta(p, a_i)$
- For any $q \in X$,
  $\alpha^X_{p,r} = \alpha^{(X-\{q\})}_{p,r} + \alpha^{(X-\{q\})}_{p,q} \cdot \left(\alpha^{(X-\{q\})}_{q,q}\right)^* \cdot \alpha^{(X-\{q\})}_{q,r}$

(in blw didn't happen exactly 1 loop.)

The desired regular expression is given as follows:

$\sum_{p \in I} \sum_{q \in F} \alpha^Q_{p,q}$. (Proof follows this sketch under the claim that $\alpha^{p,q}_{p,q}$ = some language)

automaton → regex $O(4^n)$



Compute $\alpha_{p,p}^{p,q,r}$ as

$$\alpha_{p,p}^{p,q,r} = \alpha_{p,p}^{p,r} + \alpha_{p,q}^{p,r} \cdot (\alpha_{q,q}^{p,r})^* \cdot \alpha_{q,p}^{p,r}.$$

It is easy to see that

$$\alpha_{p,p}^{p,r} = 0^*, \qquad \alpha_{p,q}^{p,r} = 0^* \cdot 1$$
$$\alpha_{q,p}^{p,r} = 00(0^*), \qquad \alpha_{q,q}^{p,r} = \epsilon + 01 + 00(0^*)1$$

Hence $\alpha_{p,p}^{p,q,r}$

$= 0^* + 0^*1 \cdot (\epsilon + 01 + 00(0^*)1)^* \cdot 00(0^*)$ } exponential blowup. every app of $+$ & $\cdot$ adds exp

$$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$$
$$\alpha + \beta \equiv \beta + \alpha$$

$$\alpha + \emptyset \equiv \alpha$$
$$\alpha + \alpha \equiv \alpha$$
$$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$$
$$\epsilon\alpha \equiv \alpha\epsilon \equiv \alpha$$
$$\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$$
$$(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$$
$$\emptyset\alpha \equiv \alpha\emptyset \equiv \emptyset$$

$$\epsilon + \alpha\alpha^* \equiv \alpha^*$$
$$\epsilon + \alpha^*\alpha \equiv \alpha^*$$
$$\beta + \alpha\gamma \leq \gamma \Rightarrow \alpha^*\beta \leq \gamma$$
$$\beta + \gamma\alpha \leq \gamma \Rightarrow \beta\alpha^* \leq \gamma$$

laws are sound and complete

soundness: $\alpha \equiv_A \beta \Rightarrow L(\alpha) = L(\beta)$

completeness: $L(\alpha) = L(\beta) \Rightarrow \alpha \equiv_A \beta$

Useful Derived Identities

$$(\alpha\beta)^*\alpha \equiv \alpha(\beta\alpha)^*$$
$$(\alpha^*\beta)^*\alpha^* \equiv (\alpha + \beta)^*$$
$$\alpha^*(\beta\alpha^*)^* \equiv (\alpha + \beta)^*$$
$$(\epsilon + \alpha)^* \equiv \alpha^*$$
$$\alpha\alpha^* \equiv \alpha^*\alpha$$

RE → ? $|A(K)| = ?$

$$\begin{aligned}
& 0^* + 0^*1(\epsilon + 00^*1)^*000^* \\
& \equiv 0^* + 0^*1(00^*1)^*000^* && \text{by (9.17)} \\
& \equiv \epsilon + 00^* + 0^*10(0^*10)^*00^* && \text{by (9.10) and (9.14)} \\
& \equiv \epsilon + (\epsilon + 0^*10(0^*10)^*)00^* && \text{by (9.8)} \\
& \equiv \epsilon + (0^*10)^*00^* && \text{by (9.10)} \\
& \equiv \epsilon + (0^*10)^*0^*0 && \text{by (9.18)} \\
& \equiv \epsilon + (0 + 10)^*0 && \text{by (9.15).}
\end{aligned}$$

Extended regular expression (ERE) ← doesn't add ↓ (Reg → DFA → Reg)

$\emptyset \mid \epsilon \mid a \mid r_1 + r_2 \mid r_1 \cdot r_2 \mid r^* \mid r_1 \cap r_2 \mid \tilde{r}$

sere (semi-extended re) (extended re)

"succinctness": descriptional complexity (ERE → RE : $n \mapsto 2^{2^{2^n}}$)

Given DFA $M = (Q, \Sigma, \delta, [q_0], F)$ and $\approx$ as before, the Quotient automaton is $M/\approx \stackrel{\text{def}}{=} (Q', \Sigma, \delta', [q_0], F')$, where

$$Q' = \{[p] \mid p \in Q\}$$
$$\delta'([p], a) = [\delta(p, a)]$$
$$F' = \{[f] \mid f \in F\}$$

Well-formedness

Lemma (Congruence) $p \approx q \Rightarrow \forall a \in \Sigma.\ \delta(p, a) \approx \delta(q, a)$.

Lemma $p \in F \Leftrightarrow [p] \in F'$

Lemma $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$.

Proof: By structural induction on x,

Algorithm [Hopcroft 1971]

1. Make pairs table with $(p, q) \in Q \times Q$ and $p \leq q$. ✓
2. → Mark (p, q) if $p \in F \wedge q \notin F$ or vice versa. ✓ phase
3. Repeat following steps until no change occurs.
   1. Pick each unmarked state (p, q).
   2. If $(\delta(p, a), \delta(q, a))$ is marked for some $a \in \Sigma$ then mark (p, q).
4. For each pair, $p \approx q$ **iff** (p, q) is unmarked.

Termination: In each pass at least one new pair must get marked.

Theorem (p, q) is marked
iff $\exists x \in \Sigma^*.\hat{\delta}(p, x) \in F \wedge \hat{\delta}(q, x) \notin F$ or vice versa.
iff $p \not\approx q$.

We prove (p, q) is marked implies
$\exists x \in \Sigma^*.\hat{\delta}(p, x) \in F \wedge \hat{\delta}(q, x) \notin F$ or vice versa.

Proof: By Ind. on phase number
Base step: $p \not\equiv_0 q \Rightarrow$ ϵ separates (p, q)
Ind. step: $p \not\equiv_{k+1} q \Rightarrow \exists a, p', q'.\ (p, q) \Rightarrow (p', q')$ and $p' \not\equiv_k q'$

$\exists y.\ \hat{\delta}(p', y) \in F \ \&\ \hat{\delta}(q', y) \notin F$ or W.

$\Rightarrow \hat{\delta}(p, ay) \in F \ \&\ \hat{\delta}(q, ay) \notin F$ or W

We prove $\exists x \in \Sigma^*.\hat{\delta}(p, x) \in F \wedge \hat{\delta}(q, x) \notin F$ or vice versa.
implies (p, q) is marked

Assume x separates (p, q) & x is shortest
Proof by Ind. on $|x|$
Base step $x = \epsilon \Rightarrow p \not\equiv_0 q$
Ind. step $(x = ay)$
x sep. $(p, q) \Rightarrow \hat{\delta}(p, ay) \in F \ \&\ \hat{\delta}(q, ay) \notin F$ or W
$\Rightarrow \hat{\delta}(\delta(p, a), y) \in F \ \&\ \hat{\delta}(\delta(q, a), y) \notin F$ W
$\Rightarrow \delta(p, a), \delta(q, a)$ is marked {Ind. Hyp}
$\Rightarrow (p, q)$ gets marked in next phase