1-1/ effective math $\sum_{i=0}^{20} z^i$ vs $\sum_{i=0}^{\infty} z^i$

$$4x^3y^2z + 8xyz - 99xy^2z^4 = 0$$

which statements are true?

"All birds have wings"

"$1+1=2$" vs "$1+1=3$"

Defining the set of true stmts

Making a decision procedure

Generating a list

A statement is a string of characters from an alphabet
some finite set Σ

A finite set is one where you can write down all the elements: $S = \{$Pikachu, Charmander, Squirtle, Bulbasaur$\} = \{C, P, B, S\}$

A string of Σ is a finite sequence of Σs

PPPP CSCSCSSB $\quad = \epsilon$

1-3/ A language is a set of strings

{ ε, P, PP, PPP, PPPP } – finite

{ ε, P, PP, ... , P^{17}, ... P^{256}, ... } – infinite

$x \in S$ – x is inside S $P \in \{ \varepsilon, P, PP \}$

$x \in X \cup Y$ iff $x \in X$ or $x \in Y$

$x \in X \cap Y$ iff $x \in X$ and $x \in Y$

$x \in \overline{Y}$ iff $x \notin Y$ (but $x \in \mathcal{U}$ – universe)

→ complement or negation of Y

$x \circ y$ = the sequence of x, then y

$PP \circ BC = PPBC$

$xy \in X \circ Y$ iff $x \in X$ and $y \in Y$

$PB \in \{P, PP\} \circ \{B, S, C\}$

$PPC \in$

is a sequence

lexicographic ordering of Σ

$lo(\{0,1\}) =$ ε, 0, 1, 00, 10, 01, 11,

000, 001, 010, 011, 100, 101, 110, 111,

...

8-1 = 7-2 = 5-4 = 1

$(\Sigma = \{0,1\}$

1-3) lexi : num $\Rightarrow$ string of Σ $\quad |\Sigma| = 2$)

lexi 0 = ε lexi 1 = 0 lexi 2 = 1

lexi 3 = 00

lexi n = $\quad sz$ = size of Σ

if $n < sz^0$ then ret ε often!

$(n - sz^0) < sz^1$ then convert $(n-sz^0)$ into a string

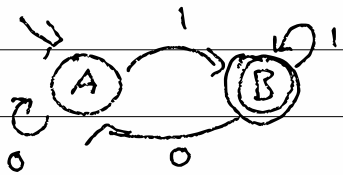
$(n-sz^0) - sz^1 < sz^2$ convert of len 2

The set of strings in the lexicographic ordering of Σ = Σ^*

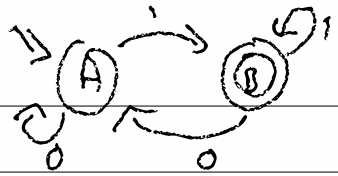
$\varepsilon \in \Sigma^*$ PRCBSPIPP $\in \Sigma^*$

0101111 $\in \Sigma^*$

Deterministic Finite Automata (DFA)



2-1) DFA



◎ means Yes

○ means No

0110 = No 1 = Yes

0111 = Yes 11 = Yes

0010 = No 00 = No

1101 = Yes

ε = No

transition function (edges) $Q \times \Sigma \Rightarrow Q$

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

always finite; are the states | | start state = A | accepting states

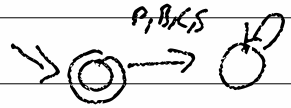
	A	B
0	A	A
1	B	B

{A, B} {0, 1} {B}

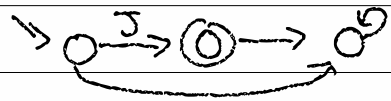
"n % 2 == 1" i.e. "odd? n"

No string DFA:

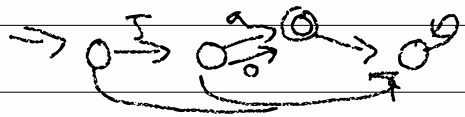
only empty string:



only the string 'J':



'Ja' and 'Jo'



$: Q \times \Sigma \to Q$

2-3/ DFA $d = (Q, \Sigma, q_0, \delta, F)$

accepts? d s : DFA × Σ^* ⇒ Bool

accepts? d ε = is q_0 in F?

d.F.member (d.q_0)

accepts d (c :: s)

: DFA : Q : Σ^*

accepts d s = helper d d.q_0 s

helper d q_i ε = $q_i \in$ d.F

helper d q_i (c::s) = helper d q_j s

q_j = d.δ(q_i, c)

```
DFA :: Accepts (3D4String s) {
    Q qi = this.q0;
    while (s != empty) {
        qi = this.delta (qi, s.first);
        s = s.rest }
    return this.F.member^(qi) }
```

↙ trace

2-3/ 0110 ⇒ Even, Odd, Odd, Even

Transducers are DFAs where trace matters

Moore machines

$L(d)$ = the language of DFA d

$= \{ s \mid \text{accepts } d\ s = \text{true} \}$

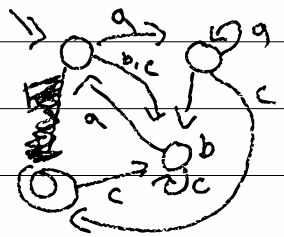
may be infinite

Given a DFA, return a string that would be accepted

example : DFA ⇒ Σ^* or false

s.t. if example d returns s then

accepts? $d\ s$ = true

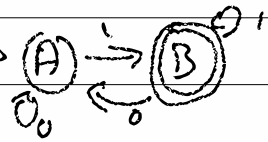


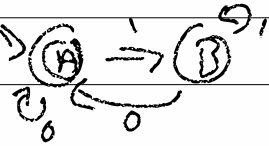
2-4/ Suppose had d in a DFA, construct d' where

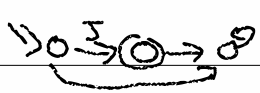
$L(d') = \overline{L(d)}$ (i.e. d' says yes when d says no & vice versa)

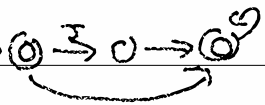
negate : DFA → DFA

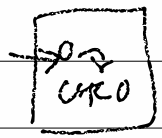
negate (Odds) = Evens

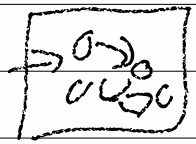
in: 

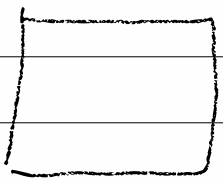
out: 

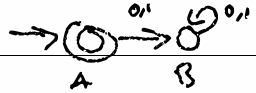
in: 

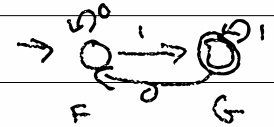
out: 

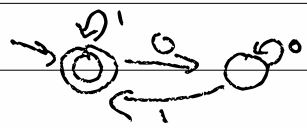
X: 

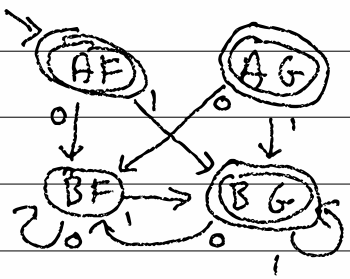
Y: 

X ∪ Y: 

X: 

Y: 

X ∪ Y: 

Z: 

2-5/ union (x : DFA) (y : DFA) = (z : DFA)

$z.Q = (x.Q \times y.Q)$

$z.\Sigma = x.\Sigma = y.\Sigma$

$z.q_0 = (x.q_0, y.q_0)$

$z.F = \{ (q_x, q_y) \mid q_x \in x.F$ or $q_y \in y.F \}$

and to make intersect

$z.\delta((q_x, q_y), c)$

$= (x.\delta(q_x, c), y.\delta(q_y, c))$

$X \subseteq Y$ (subset) iff

$\forall q \in X.\ q \in Y.$

$X = Y$ iff

$X \subseteq Y$

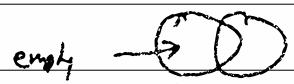
and $Y \subseteq X$

subset? : DFA × DFA ⇒ bool

subset? (≫ϕ) X = Yes

(epsilon) (Evens) = Yes

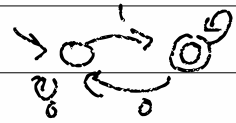
(epsilon) (odds) = No



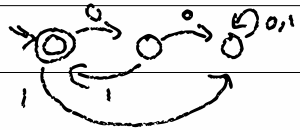
$X - Y$ must be empty

$X \cap \overline{Y}$ if empty

3-1 $z \in X \circ Y$ iff $z = x \circ y$ where $x \in X$ and $y \in Y$

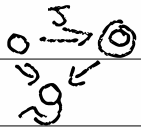
$X =$ 

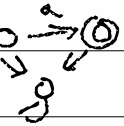
$x \in X = 0111$

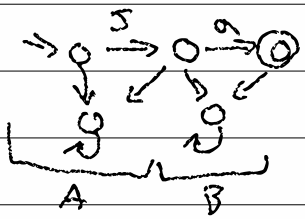
$Y =$ 

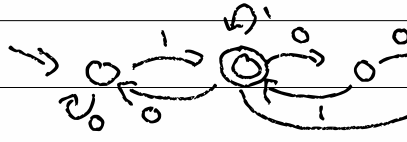
$y \in Y = 010101$

$z \in X \circ Y =$ 0111 010101 (x, y)

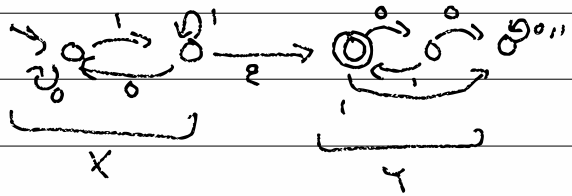
$A =$ 

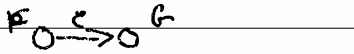
$B =$ 

$A \circ B =$ 



// illegal, not a DFA!





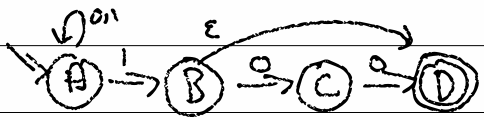
(skip from F to G at the right time)

3-2/ NFA - non-deterministic finite automata

DFA = $(Q, \Sigma, q_0, \delta : Q \times \Sigma \Rightarrow Q, F \subseteq Q)$

NFA = $(Q, \Sigma, q_0, \delta : Q \times \Sigma_{\text{or } \varepsilon} \to P(Q), F \subseteq Q)$

(sigma, epsilon)



δ	A	B	C	D
0	{A}	{C}	{D}	∅
1	{A,B}	∅	∅	∅
ε	∅	{D}	∅	∅

(A,0) (A,1) (A,0) (A,0)

x ∉ L(n) iff

∀t. oracle n t = #f

(or "if")

trace = a sequence of $Q \times \Sigma$ or ε

(A,0) (B,1) (C,0) (D,0) 0100 ∈ L(n)

(A,1) (A,0) (B,1) (D,ε) 101ε = 101 ∈ L(n)

oracle interpretation : NFA x trace → boolean

oracle N t = helper N N.q0 t

helper N q_i [] = $q_i \in N.F$

helper N q_i (q_i, c) :: t') =

is $q_j \in N.\delta(q_i, c)$, then helper N q_j t'

o.w. "invalid trace"

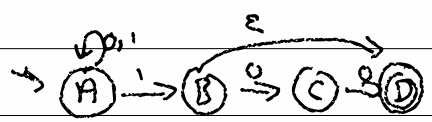
3-3/ trace-tree = accept | reject
| branch state (tt set)

all : NFA × Σ^* → tt

all N s = helper N $N.q_0$ s

helper N q_i s =
branch q_i (case s where
[] → if $q_i \in N.F$ then
{accept}
o.w
{reject}

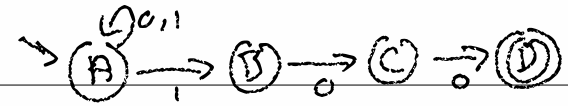
c :: s' →
{ tt | tt = helper N q_i s'
where $q_i \in N.\delta(q_i, c)$ }
∪ { tt | tt = helper N q_i s
where $q_i \in N.\delta(q_i, \varepsilon)$ and $q_i \neq q_i$ }



0100

0 1 A → A → A → Reject

A → A
1 → B → C → D → ACCEPT
= (branch D (accept))
ε → D
= (branch D ∅)

3-4/ 

"ends in 100"

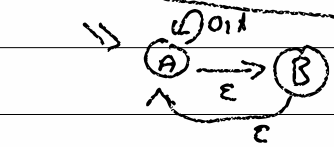
011100

A -0-> A -1-> A -1-> A -1-> A -0-> A -0-> A -> REJ
A -1-> B -1-> B -1-> B -0-> C -0-> D -> ACC

backtrack : NFA × String Σ^* → Bool
backtrack N s = helper N N.q_0 s
helper N q_i s =
  OR (case s with [] → $q_i \in N.F$
    c::s' → OR helper N q_j s')
      $q_j \in N.\delta(q_i, c)$

OR helper N q_j s
$q_j \in N.\delta(q_i, \varepsilon)$

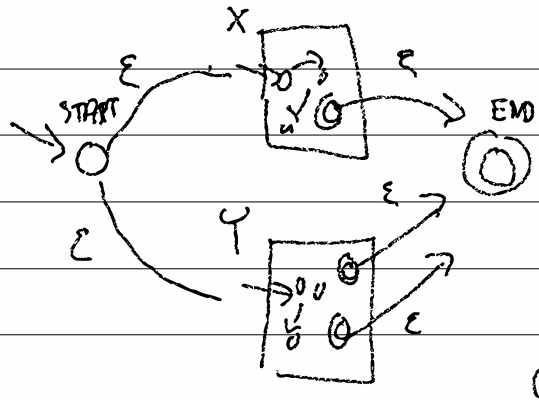
x=3 ; (1 || x++) ; x==3,



maybe DS = tree

unfold : A → DS(B)
fold : DS(B) → C Haskell
  these always exist
combined : A → C

4-1



$X = (Q_x, \Sigma, q_{0x}, \delta_x, F_x)$

$Y = (Q_y, \Sigma, q_{0y}, \delta_y, F_y)$

$Z = (Q_z, \Sigma, q_{0z}, \delta_z, F_z)$

$F_z = \{END\}$

$q_{0z} = \{START$

$Q_z = \{START, END\}$

$\delta_z(q_i, c) =$

$(START, \varepsilon) = \{(q_{0x}, 0)\ (q_{0y}, 1)\}$

$(START, \_) = \{\}$

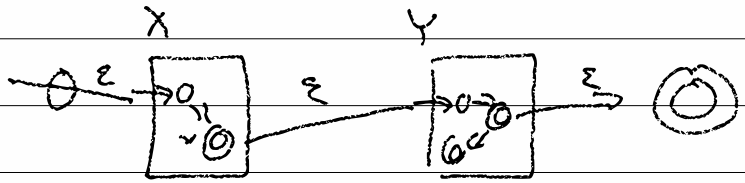
$(END, \_) = \{\}$

$((q_x, 0), c) = \delta_x(q_x, c) \times \{0\}$

q_y, similar $\cup$ if $q_x \in F_x$ and $c = \varepsilon$, $\{END\}$ o.w. $\{\}$

$\cup\ Q_x \times \{0\}$

$\cup\ Q_y \times \{1\}$



4-2/ Kleene-star X^*

$z \in X^*$ iff $z = \varepsilon$ OR $z = xy$

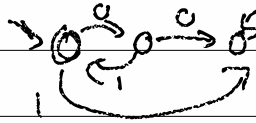
where $x \in X$ and

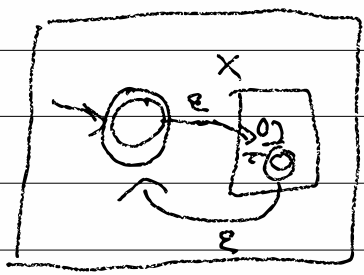
$y \in X^*$

iff $z = x_0 \dots x_n$

where $x_i \in X$

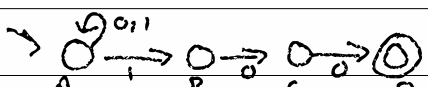
$(\{0\}\{1\})^*$ = any number of 01 sequences



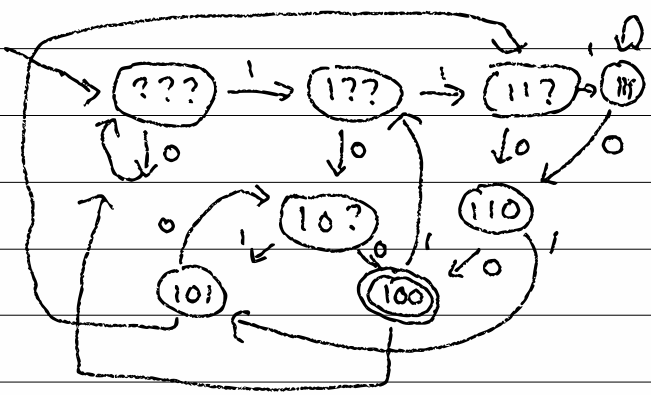


DFAs : $\cup$, $\cap$, $\overline{\ }$ DFA → NFA

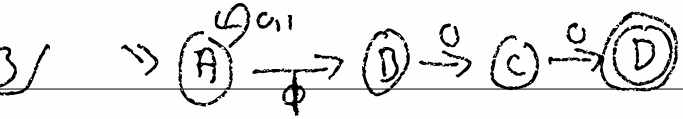
NFAs : $\cup$, $\circ$, $*$ want: NFA → DFA

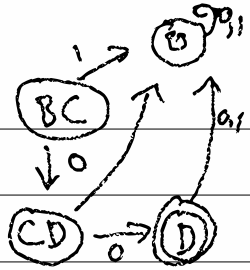


01100



4-3/



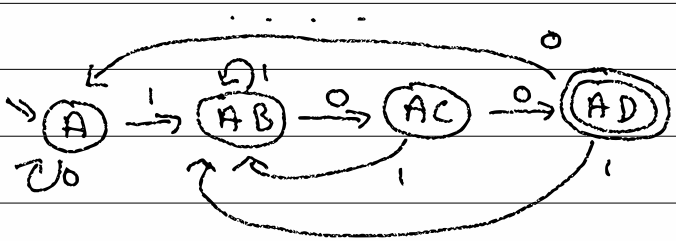


01100

A → A → A → A → A → A

B B C D ← ✓

??? ??? 1?? 11?? 110? 100



01100 ✓

011001 X

1111 X

111100 ✓

NFA ⇒ DFA in: $(Q_N, \Sigma, q_{0N}, \delta_N, F_N)$

out: $(Q_D, \Sigma, q_{0D}, \delta_D, F_D)$

$Q_D = P(Q_N)$ $\quad q_{0D} = E(\{q_{0N}\})$ $\quad F_D = \{q_d \mid F_N \cap q_d \neq \emptyset\}$

$$\delta_D(q_d, c) = E\left(\bigcup_{q_n \in q_d} \delta_N(q_n, c)\right)$$

$E: Q_D \Rightarrow Q_D$ $\quad E(q_d) = q_d \cup \bigcup_{q_n \in q_d} \delta_N(q_n, \varepsilon)$

4-4)

$\forall x \in \Sigma^*$. ~~accepts~~ backtrack N x

= accepts (NFA→DFA N) x

random string

= pick a random number

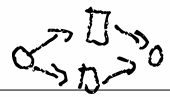
(lexi n)

write the NFA manually as a DFA...

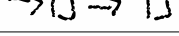
use the dFA equality checker to see that

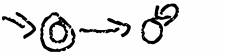
dfa-equal? manual (N→D N) = #t true

5-1/ regular expression := $x \cup y$ / $x \cap y$ / $x \circ y$ / $\bar{x}$ / x^* / $\emptyset$ / ε / $c \in \Sigma$ ← Then regular operations

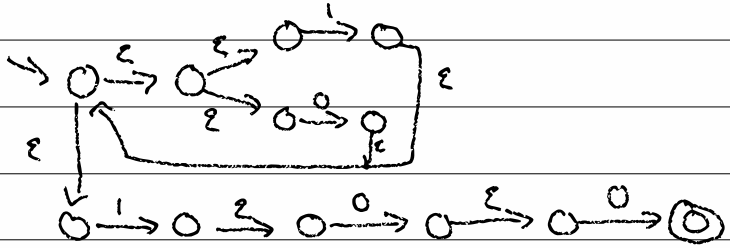


x, y Lre

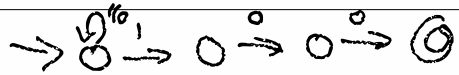




$('1' \cup '0')^* \circ '1' \circ '0' \circ '0' = (1 \cup 0)^* 100$



=



$\#_1(w) \equiv 0 \pmod 2$

$\#.c \Rightarrow \Sigma^* \circ '.' \circ 'c'$

5-2/ interface RegEx;

class	Empty	implements RegEx;	()
	Epsilon	RegEx	()
	Char	RE	(char c)
	Union	RE	(RE x, RE y)
	Star	RE	(RE x)
	Concat	RE	(RE x, RE y)

```
new Concat ( new Star (new Union (new Char('1'),
                                  new Char('0'))),
             new Concat ( new Char ('1'),
                          new Concat (new Char ('0'),
                                      new Char('0'))))
```

=

$(1 \cup 0)^* 100$

```
interface RegEx { NFA compile(); }

Union::compile() { return nfa_union(this.x.compile(),
                                    this.y.compile()); }
```

5-3/ generate : RE → Σ^* s.t.

accepts? (nfa→dfa(compile r)) (generate r) = true

generate ∅ = error

gen e = ε

gen c = c

generate x○y = gen x ○ gen y

gen x^* = gen (ε ∪ x○x^*)

gen x∪y = case (flip)

heads ⇒ gen x

tails ⇒ gen y

printall : RE ⇒ void

printall r = helper r print

helper ∅ pr = ~~err~~ (void)

h ε pr = pr ε

h c pr = pr c

h x○y pr = h x new-print

(new-print s = h y np2

np2 t = pr s○t)

= h x (lambda s: h y (lambda t: pr s○t))

h x^* pr = h (ε ∪ x○x^*) pr

h (x∪y) pr = h x pr ; h y pr

numbers

S-Y $X \cdot 0 = 0$ $\quad X + y = y + X$ $\quad X \cdot 1 = X$

regex

$X \circ \emptyset = \emptyset = \emptyset \circ X$ $\quad \emptyset = \{\}$

$X \circ \varepsilon = X = \varepsilon \circ X$ $\quad \varepsilon = \{\varepsilon = ""\}$

$\emptyset \cup X = X = X \cup \emptyset$

$X \cup (X \cup y) = X \cup y$

$\emptyset^* = \varepsilon$ $\quad X^* = \varepsilon \cup X \circ X^*$

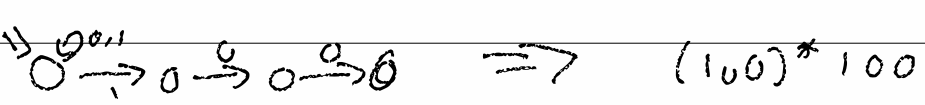
$\varepsilon^* = \varepsilon$ $\quad = \emptyset \circ \emptyset^* \cup \varepsilon = \varepsilon \cup \emptyset = \varepsilon$

$(X^*)^* = X^*$ $\quad \varepsilon^* = \varepsilon \cup \varepsilon \circ \varepsilon^*$

$\varepsilon \cup \varepsilon^*$

$Z \supseteq X$ if $X \cup Z = Z$

DFAs $\iff$ NFAs



IN

G-1/ NFA(k) → GNFA (2+k)

RIP [→ GNFA (2+k-1) → GNFA (2+k-2) ...] — k times

→ GNFA (2)

OUT → REX

NFA: $0 \xrightarrow{\varepsilon} 0$

$0 \xrightarrow{a} 0$

GNFA: $0 \xrightarrow{r} 0$

GNFA (generalized NFA)

$= (Q, \Sigma, q_a, \Delta, q_f)$

↑ one state, not a set

$\Delta : (Q \times Q) \to Rex$ $\qquad \delta : Q \times \Sigma \Rightarrow P(Q)$

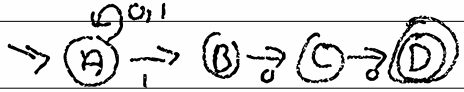
$(Q - q_f)$ $(Q - q_a)$

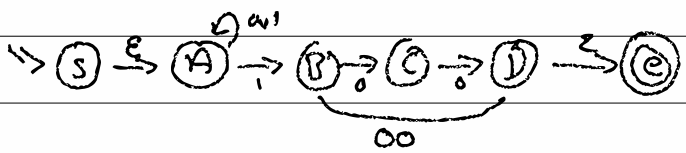
You can't leave q_f — You can't go back to q_a

(S) $\xrightarrow{r}$ (e)

if

$\Delta(q_i, q_j) = r$ ~~if~~ $x \in r$ then $q_i \xrightarrow{x} q_j$ in the NFA

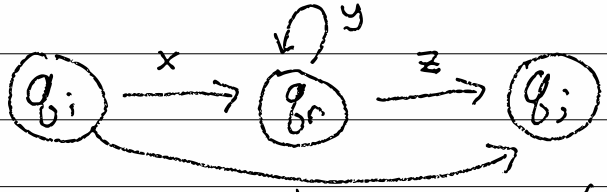




6-2/ $R_{io} : GNFA(n+1) \Rightarrow GNFA(n)$

$\{q_0, q_f, q_r, q_i \dots\} \Rightarrow \{q_0, q_f, q_i \dots\}$

q_r is gone



$(x \circ y^* \circ z) \cup d$

...

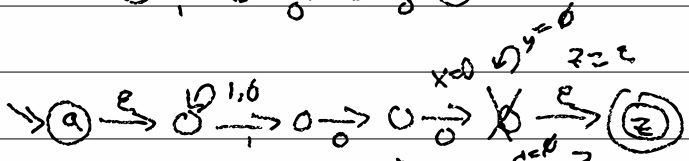
$x \circ y^* \circ z \cup d$

$\Rightarrow$

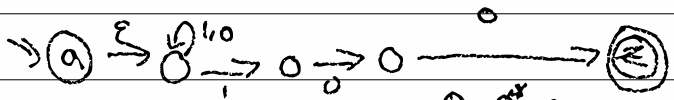
~~$x \circ z$~~ ~~$x \circ z \cup d$~~

IN

6-3/



RIP

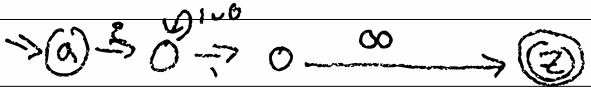


$xy^*z \cup d$

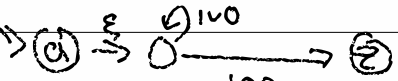
$0 \cdot \emptyset^* \circ \varepsilon \cup \emptyset$

$= 0$

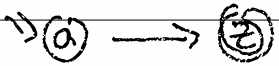
RIP



RIP



RIP

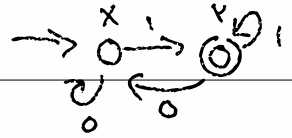


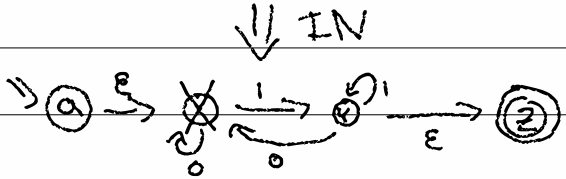
$\varepsilon \circ (1 \cup 0)^* \circ 100 \cup \emptyset$

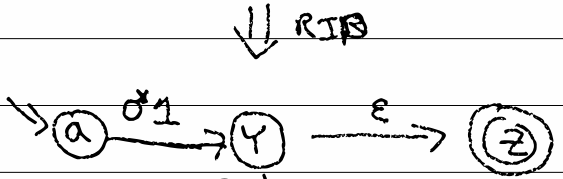
OUT

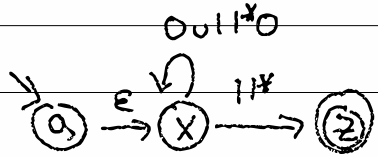
$= (1 \cup 0)^* \circ 100$ ←

6-4/









RIP

$(0 \cup 11^*0)^* 11^*$

RIP

$0^*1 \circ (1 \cup 00^*1)^*$ = //

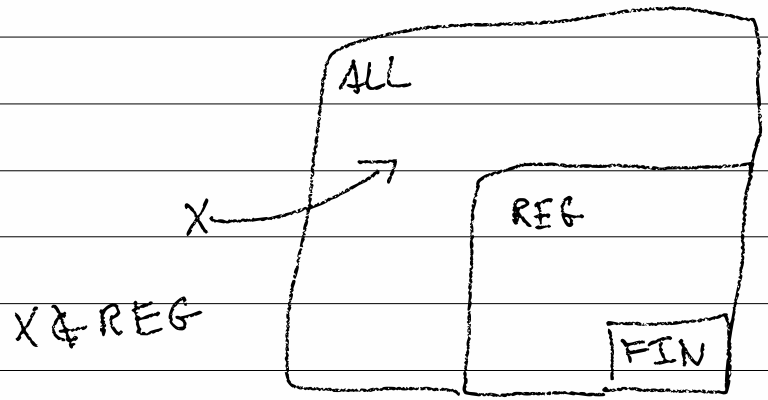
$(0 \cup 1)^* 1$

DFA $\rightleftharpoons$ REX

$\forall d.$ let r = dfa⇒rex d

accepts? d (generate r) = true

G-5



REG = language described by DFA, NFA, or RE

FIN = finite languages

X & REG

ALL = $P(\Sigma^*)$ $\quad \Sigma = \{0, 1\}$

ALL = { $\{\}$, $\quad \Sigma^* = \varepsilon, 0, 1, 00, 01, 10, 11, \ldots$

$\{0, 1\}$, $\{0, 00, \varepsilon, 000, 0000, \ldots\}$

{all of Jay's lectures}

{JPEGs of cats}, {JPEGs of road signs}

... }

REG = ALL?

Z-1/ $\exists X \in ALL . \; X \notin REG$

- $\exists$: language (some problem)
- $X \in ALL$: all possible languages
- $X \notin REG$: languages NOT recognized by DFAs

option 1: $\forall Y \in REG. \; X \neq Y$

option 2: $\forall Y \in REG. \; P(Y)$
$\neg P(X)$

mystery #1: What is X? witness
#2: what is P? property

$\Rightarrow$

proof 1: $\forall Y \in REG . \; P(Y)$
proof 2: $\neg P(X)$

$\Rightarrow$

$X \in ALL$, but $X \notin REG$

$\Rightarrow$

computers aren't omnipotent

7-2/ DFAs .. $(Q, \Sigma, q_0, \delta, F)$

alphabet

finite set of states

start somewhere

many stop

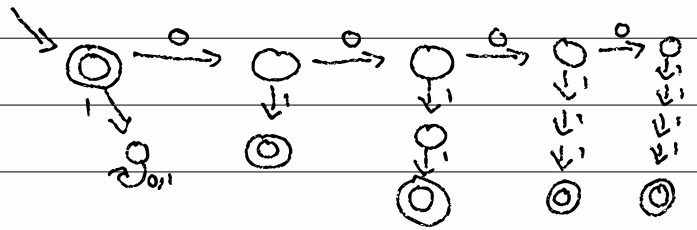
$Q \times \Sigma \to Q$

EQ

$\varepsilon \in EQ$ $\quad 01 \in EQ$ $\quad 0011 \in EQ$

$0 \notin EQ$ $\quad 010 \notin EQ$ $\quad 000111 \in EQ$

$0110 \notin EQ$ $\quad 0000\ 1111 \in EQ$



$= EQ^4$

$|EQ^0| = 2$ $\quad |EQ^1| = 4 = |EQ^0| + 2$ $\quad |EQ^3| = 11 = |EQ^2| + 4$

$|EQ^2| = 7 = |EQ^1| + 3$ $\quad |EQ^n| = |EQ^{n-1}| + n + 1$

$= \frac{(n+1)(n+2)}{2} + 1$

$\forall n.\ 0^n 1^n \in EQ$

$\wedge\ 0^n 1^n \in EQ^m$ where $n \le m$

$\forall m.\ \exists n.\ 0^n 1^n \notin EQ^m \quad (n = m+1)$

7-3/ int count = 0; char c;

while (c = getc(); c == '0')

count++;

ungetc(c);

while (c = getc(); c == '1')

count--;

return count == 0;

$\Longrightarrow$

EQ^m, what is m? $m = 2^{31}-1$

$m = 2^{2^{33}}$

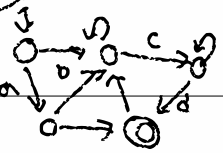
$0^n 1^n \in EQ$ for all n

$=$

$\times$

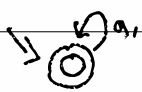
step 1: what is X? ✓

7-4/ What is P?

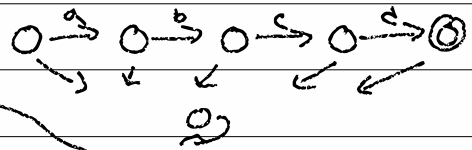


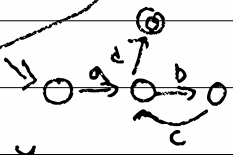
abcd ∈ X

6

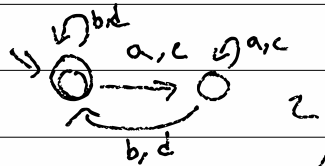


1

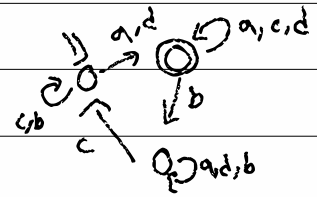




4



2



3

1: a aaa bcd ∈X

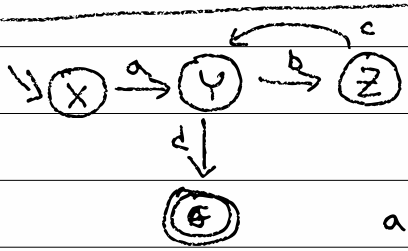
2: ababababcd ∈X

3: abcabcabcd ∈X

4: a bcbcbc d ∈X

5: x

6: x



abcd string (i=1)

X Y Z Y G trace

abcbcbcd string (i=3)

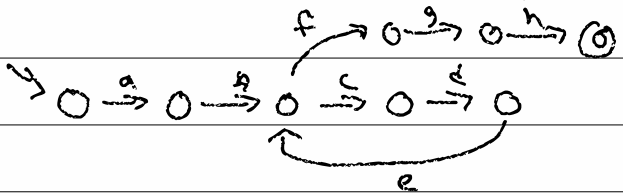
X Y Z Y Z Y Z Y G trace

$a(bc)^i d \in DFA$ for all $i \in [0, \infty)$

ad string (i=0)

X Y G

7-5/ DFAs must contain cycles!



If $s \in$ DFA, and s is long enough ($|s| \geq |Q|$),
then s must visit some state $= q_r$ twice!

ex: $s = abcd$ s visited γ twice $|Q| = 4$

We express s as $x \circ y \circ z$

where x goes from q_0 to q_r

(y isn't empty $\neq \varepsilon$) y goes from q_r to q_r

$|xy| \leq |Q|$ z goes from q_r to $q_f \in F$

ex: $x = a$ $y = bc$ $z = d$ $q_0 = X$ $q_r = Y$ $q_f = G$

That means for all i:

$$x \circ y^i \circ z \in \text{DFA}$$

ex: $i = 0$ is $ad \in$ ✓ $i = 3$ is $abcbcbcd \in$ ✓

7-6/ Regular Pumping Property (RPP)

$RPP(A : Language) ::=$

$\exists p \in \mathbb{N}.$

$\forall (s \in A \mid |s| \geq p)$

$\exists (x, y, z \in \Sigma^* \mid |xy| \leq p \wedge |y| > 0)$

$\forall i \in \mathbb{N}.$

$xy^i z \in A.$

Pumping Lemma: $\forall A \in REG.\ RPP(A).$

$p = |Q|$, x is the string before q_r

y is from q_r to q_r

z is from q_r to $q_e \in F$

step 2: What is P? ✓

step 3: $\forall A \in REG.\ P(A).$ ✓

step 4: $\neg P(EQ)$ ...

8-1) $\neg RPP(EQ)$

$= \forall p \in N.$

$\exists (s \in A \mid |s| > p)$

$\forall (x,y,z \in \Sigma^* \mid$

$|y| > 0$

$|xy| < p)$

$\exists\ i \in N$

$xy^i z \notin A$

$\neg(A \wedge B) = \neg A \vee \neg B$

$\neg(A \vee B) = \neg A \wedge \neg B$

$\neg \forall x. P(x) = \exists x. \neg P(x)$

$\neg \exists x. P(x) = \forall x. \neg P(x)$

$\forall$ = you don't choose it

$\exists$ = you DO choose it

$\neg RPP(EQ)$

given p.

choose $s \in EQ$ where $|s| > p$

$s = 0^p 1^p$

given x, y, z where $|y| > 0$ $|xy| < p$

$s = xyz$ $0^p 1^p = xyz$ $b > 0$ $a+b+c = p$

$x = 0^a$ $y = 0^b$ $z = 0^c 1^p$ $a+b < p$

choose i $(i = 0)$

$xy^i z \notin EQ$ $xy^i z = 0^a 0^{b \cdot i} 0^c 1^p \notin EQ$

iff $a + bi + c \neq p$

$a + bi + c \neq a + b + c$

$bi \neq b$

$i \neq 1$

8-2/ $S = xyz^{\in A}$ and $xy^i z \in A$

then

$xy^* z \in A$

Regular expression

$xy^* z \subseteq A$

ALL $\not\rightarrow$ REG because $EQ \in ALL$

$EQ \notin REG$

$MEQ^0 = 0^0 \circ EQ$

$MEQ^1 = 0^1 \circ EQ$

$MEQ^n = 0^n \circ EQ$

$000 \circ 0011 \in MEQ^3$