

1-1/

How do we know if a math formula  
is true?

How do we know if an algorithm  
(like Euclid's GCD) "works"?

↙ ↓

correct effective

Does an algorithm exist?

What is an algorithm?

Does a program exist? ← problems

What is a program? ← models

I-2] A set is "a bunch of stuff"

$\emptyset$  - nothin' in it  
 $\forall x, x \notin \emptyset$

$\{ \text{pen, phone} \}$      $\{ \text{phone, pen} \}$

$\{\checkmark, \square\}$

$\nexists \text{ pen} \in \{ \text{pen, phone} \}$

$\forall x, x \in \{y\}$  iff  $x = y$

union -  $\cup$        $A \cup B$

$\forall x, x \in A \cup B$  iff  $x \in A$  or  $x \in B$

$\{ \text{pen, phone} \} = \{ \text{pen} \} \cup \{ \text{phone} \}$

[=3] "The set of all true math formulas"

A set IS its membership

" $1+1=2$ "  $\in TS \uparrow ?$

"Is there a god?"

"Will Buffy be remade?"

All sets "constructed" via  $\emptyset$ ,  $\{\cdot\}$ ,  $\cup$  are finite.

$$x \in \{\emptyset\} \cup \{\{\cdot\}\}$$

The Universe ( $U$ )

$A \subseteq B$  iff  $\forall x, x \in A \rightarrow x \in B$

↳ Our universe is made of strings  
and strings are sequences of characters  
and chars are elements of an alphabet  
an alphabet is a finite set



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

↑  
chars

$$\{0, 1, \cup, \$, +\}$$

↑  
chars

"0100001" = a string = s

length = 7

$$s(0) = 0$$

$$s(1) = 1 \quad s(2) = 0$$

U =  $\Sigma^*$  ← special notation

$$A^* = \{\epsilon\} \cup A \circ A^*$$

epsilon = " " = the string w/ no characters

$x \in A \circ B$  iff  $x(0) \in A$  and  
 $x(1..) \in B$

$$\{0, 1\} \circ \{0, 1\} = \{00, 01, 10, 11\}$$

$$\{1\} \circ \{0\} = \{10\}$$

LS / #1. Decide a data type to represent alphabets and characters.

Alphabet = List < Character >

Character = Object / void\*  
we need equality

#2. Decide a data type for strings

interface String { }

class MtString implements String { .. }

class OneString impl String { }

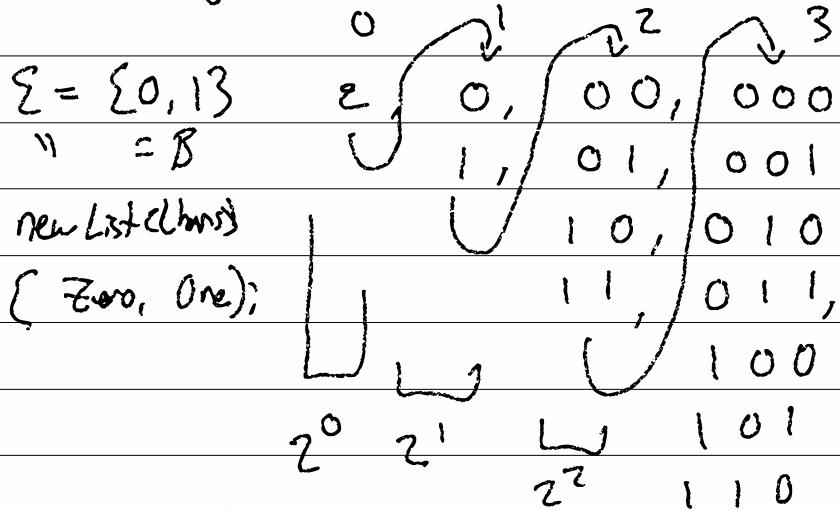
OneString ( char c, String s ) { ... }

Zero = new BasicChar('0'); One = new BC('1');

010 = new OneS(Zero, new OneS(One, new  
OneS(Zero, new MtS())));

(EP)

1-6) Every alphabet has a lexicographical ordering of the strings in  $\Sigma^*$



$|A|^i$  where  $i$  is layer  $2^i$   
looking at

lexi :  $\Sigma \times \mathbb{N} \rightarrow \Sigma^*$

lexi  $\emptyset 0 = \emptyset$

lexi  $\emptyset 1 = 0$

lexi  $\emptyset 2 = 1$

lexi  $\emptyset 6 = 101$

2-1 "1+1"  $\rightarrow$  "2"

"1+1 = 2"  $\in$  Truth

"1+1 = 3"  $\notin$  Truth

$\emptyset \quad \Sigma^3 \quad A \cup B$

Alphabet  $\Sigma$  Universe  $\Sigma^*$

{0, 1}

{ε, 0110, 000001,



3

$P(A) \quad 2^A$

$x \in P(A)$  iff  $x \subseteq A$  ( $x \subseteq A$ , iff  
 $\forall y \in x, y \in A$ )

$A = \{0, 1, 2, 3\}$

$\emptyset \in P(A) \quad \emptyset \subseteq A$

0110 {1, 2}

{0}  $\in$   $\emptyset \subseteq \{2, 3\} \in \{0, 1, 2, 3\} \quad \underline{\textcircled{1}} \quad 123$

$P(\Sigma^*) \quad \Sigma^* = \{ \epsilon, 0, 1, 00, 111111 \}$

$\emptyset \in P(\Sigma^*)$

...

{ε}  $\in P(\Sigma^*)$

0011, ...

all even length strings  $\in P(\Sigma^*) = \{ \epsilon, 00, 11, 01, \dots \}$

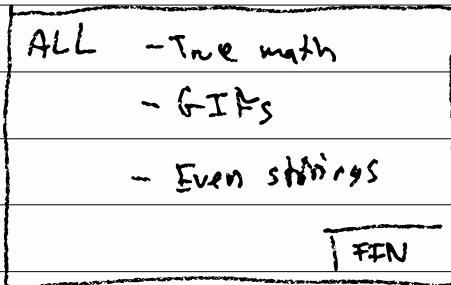
GIFS  $\in P(\Sigma^*)$

{GIFS of me}  $\in P(\Sigma^*)$

JPGs w/ a cat in them  $\in P(\Sigma^*)$

2-2  $\text{ALL} = \mathcal{P}(\Sigma^*)$

$\text{FIN} =$  the set of  
finite sets



$\emptyset \in \text{FIN}$

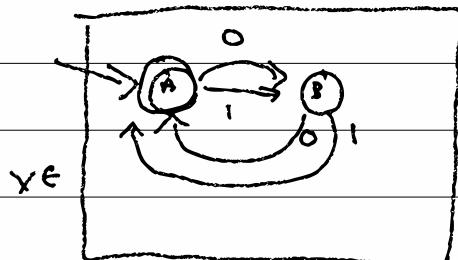
$\forall x \in \Sigma^*, \{x\} \in \text{FIN}$

$A \in \text{FIN} \wedge B \in \text{FIN}$

$\Rightarrow A \cup B \in \text{FIN}$

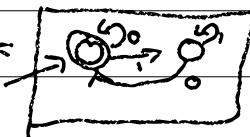
All even strings =

DFA - a deterministic finite automata



$\Sigma$  or  $\Sigma^*$

even nums =



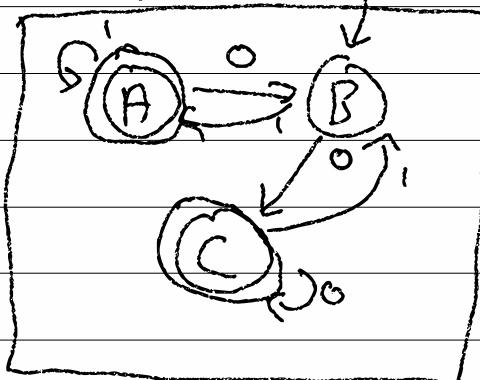
○ - states  $\Sigma A, B \}$

$\rightarrow \circlearrowleft$  - start state  $A$

$\odot$  - accepting states  $\Sigma A \}$

$\overset{x}{\rightarrow} \circlearrowright$  - transition  $\Sigma$

$x$  - labels are  $\Sigma$



$\Sigma A, B, C \}$

$\Sigma A, C \}$   $\Sigma = \Sigma_0, 1 \}$

A	0	1
B	C	A
C	C	B

2-3)  $x \in \text{DFA} (\underbrace{\text{states}, \text{alphabet}, \text{start}, \text{accepting}}_{\text{states } Q, \Sigma, q_0 \in Q, F \subseteq Q}, \delta: Q \times \Sigma \rightarrow Q - \text{transitions})$

DFA configuration =  $Q \times \Sigma^*$   
 $\stackrel{\uparrow}{[q]} w^{\uparrow}$

config update function : config  $\times \text{DFA} \rightarrow \text{config}$   
 $[q]w \rightarrow [q']w'$

$[q_i]x \rightarrow [q_j]y \text{ iff } \delta(q_i, x) = q_j$   
 $x \in \text{DFA} \text{ iff } [q_0]x \Rightarrow \Rightarrow \Rightarrow \Rightarrow [q_f] \in$   
 and  $q_f \in F$

0110  $\in \text{EvenLen}$  ;iff  $[A]0110 \rightarrow [B]110 \rightarrow [A]10$   
 $\rightarrow [B]0 \rightarrow [A] \in AF\{A\}$  ✓

class DFA  $\Sigma$

..  $Q, \Sigma, F, q_0, \delta \dots$

public bool accepts (String x) {

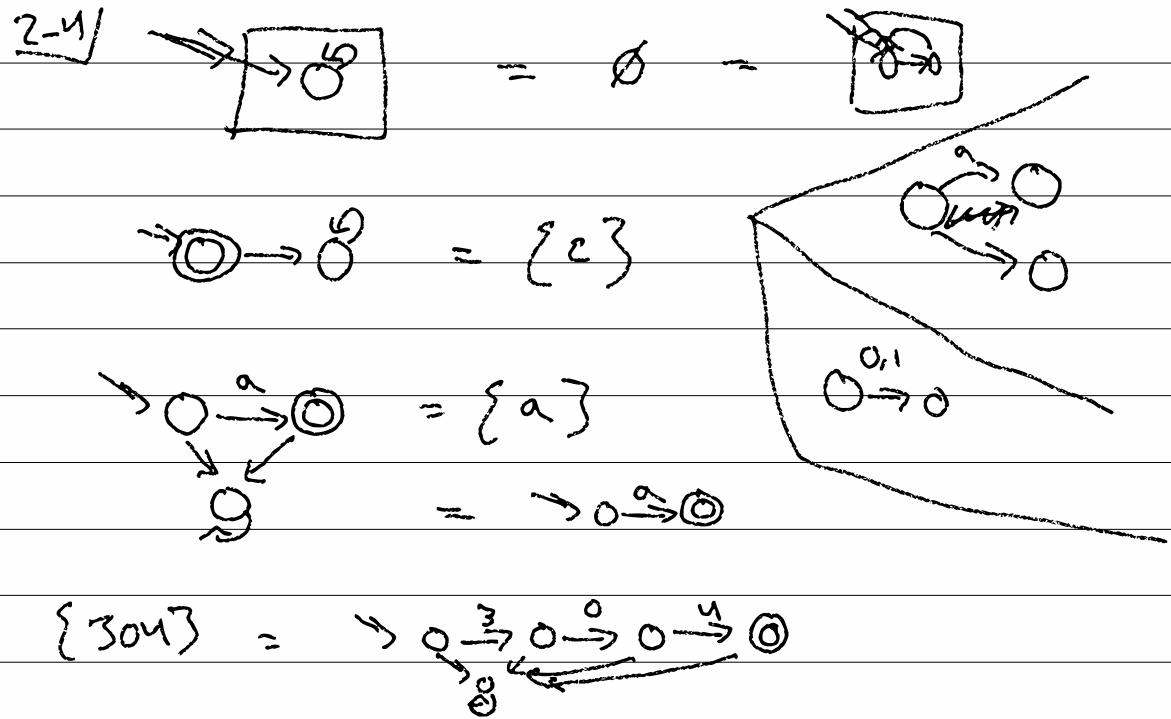
State  $q_i = q_0;$

while ( $(x, \cancel{\neq} \text{empty}) \Sigma$

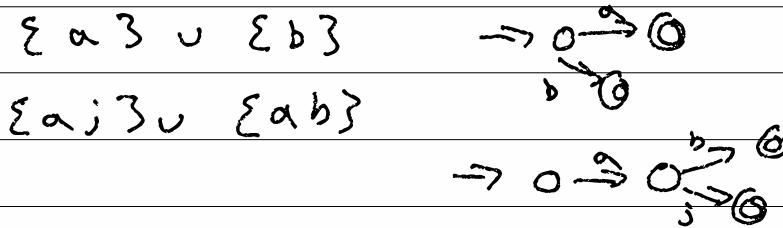
$q_i = \delta(q_i, x, \text{first}());$

$x = x, \text{rest}();$  } }

return  $F, \text{in}(q_i);$  } }



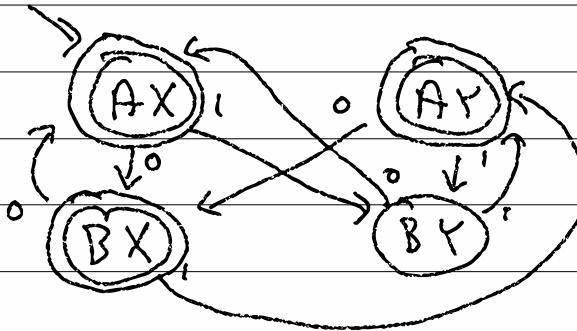
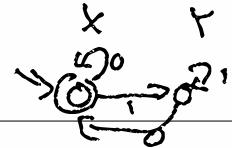
$A \cup B \leftarrow \text{DFA}$  (if  $A \in \text{DFA}$  and  $B \in \text{DFA}$ )



2-5 Even Len



Is Even



0	✓
00	✓
11	✓
0	✓
110	✓

n

$(x, y) \in A \times B$

; if  $x \in A \wedge y \in B$

$$A = (Q_A, \Sigma, g_{0A}, \delta_A, F_A)$$

$$B = (Q_B, \Sigma, g_{0B}, \delta_B, F_B)$$

$$X = A \cup B$$

$$Q_X = Q_A \times Q_B \quad \delta_X = ((g_A, g_B), c) =$$

$$g_{0X} = (g_{0A}, g_{0B}) \quad ( \delta_A(g_A, c),$$

$$F_X = F_A \times F_B - n \quad \delta_B(g_B, c) )$$

$$F_A \times Q_B \cup Q_A \times F_B - V$$

$x \in A \cap B$  ; if  $x \in A \wedge x \in B$

2-6)  $x + A^c$  iff  $x \notin A \quad (x \in u)$

Even Len      odd Len  
 $\rightarrow \textcircled{0} \rightarrow \textcircled{0}$        $\Rightarrow \rightarrow \textcircled{0} \rightarrow \textcircled{0}$

$$F = \{A\}$$

complement

$$F' = Q - F$$

or  $F^c$  (wrt Q)

Algorithm for  $X \subseteq Y$  if  $X, Y$  are DFAs

### 3-11 DFA $\Rightarrow$ example or false

DFA:

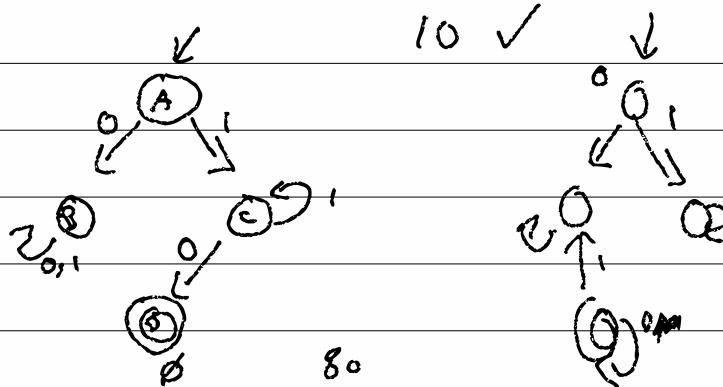
$Q$ : Sstate  $\Rightarrow$  Bool

$\Sigma$ : list of characters

$s_0$ : state

$S$ : (state  $\times$  char)  $\rightarrow$  Sstate

$F$ : State  $\Rightarrow$  Bool



$\Sigma = \{A, B, C, D\}$

$\Sigma = \{A\}$

$[A]$

$A \Rightarrow \text{?}$

$\Sigma = \{B, C, D\}$

$\Sigma = \{A\}$

$[B, C]$

$B \Rightarrow A, 0$   
 $C \Rightarrow A, 1$

$\Sigma = \{C, D\}$

$\Sigma = \{A, B\}$

$[C]$

'Yes, it is possible.'

$\Sigma = \{D\}$

$\Sigma = \{A, B, C\}$

$[D]$

$D \Rightarrow C, 0$

$\Sigma = \{E\}$

$\Sigma = \{A, B, C\}$

$[E]$

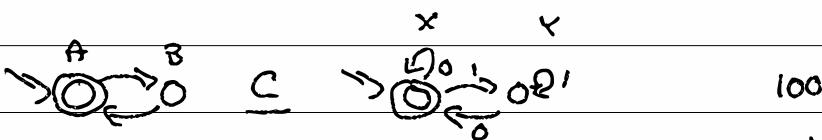
or, if not, No!

3-2/ subset

$A \subseteq B \iff \forall x \in A. x \in B \rightarrow x \in B$

$$\{\alpha, \beta\} \subseteq \{\alpha, \beta, \gamma\} \quad U = \{\alpha, \beta, \gamma\}$$

finite means naive works!



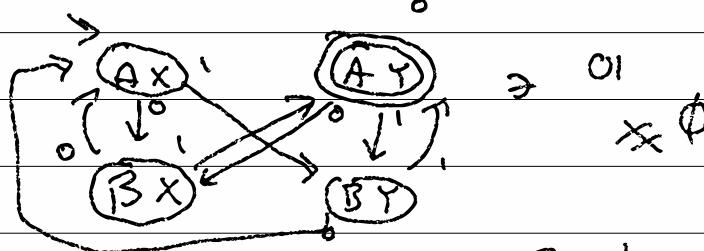
$$\boxed{\bar{A}} \subseteq \boxed{\bar{B}} \cap \boxed{\bar{A}} = \boxed{\bar{A}}$$

$$\boxed{B} \cap \boxed{\emptyset} = \boxed{\text{shaded}} \emptyset$$

Diagram illustrating set intersection and empty set:

- Set  $\bar{A}$  is shown intersecting with set  $\bar{B}$ , resulting in set  $\bar{A}$ .
- Set  $B$  is shown intersecting with the empty set, resulting in the empty set.

$$\mathbb{B} \text{ EvenNum} = \rightarrow \xrightarrow{x^0} \xrightarrow{y^0} \xrightarrow{z^1} \emptyset^2$$



soundness: model  $\subseteq$  theory  
 completeness: theory  $\subseteq$  model  
 $\text{model} = \text{theory}$

$$3-3) \quad 0, 1, 2, -1, 5 \quad \mathbb{Z}, \mathbb{P}, \mathbb{N}$$

$$\{\mathbb{P}\} + \{\mathbb{N}\} = \{\mathbb{P}, \mathbb{Z}, \mathbb{N}\}$$

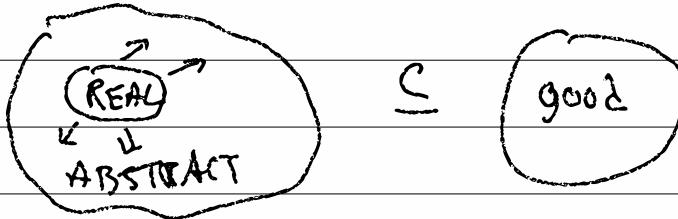
if  $x > 0$  then

$$A \quad y = 5 \Rightarrow \{\mathbb{P}\}$$

o.w.

$$B \quad y = 0 \Rightarrow \{\mathbb{Z}\}$$

$\Rightarrow$  assume  $y = \{\mathbb{P}, \mathbb{Z}\}$



<u>3-w)</u>	Finite	=	$\emptyset$	$\Sigma^3$	$A \cup B$	EDFA
			$A^c$	$A \cap B$	$A \circ B$	

Infinite =  $A^*$

\*  $x \in \Sigma^* \wedge y \in \Sigma^*$  then  $xoy \in A \circ B$  iff  
 $x \in A \wedge y \in B$

$$\varepsilon \circ y = y \quad \text{if } a \in \Sigma, (a \circ x) \circ y = a \circ (xoy)$$

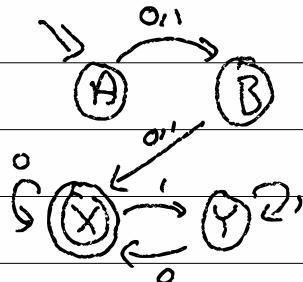
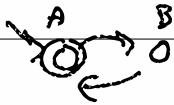
$$abcd = ab \circ cd$$

$$\{\text{jm}\} \circ \{\text{mj, nj}\} = \{\text{jim, jn}\}$$

$x \in A^*$  iff  $x = x_0 \circ x_1 \circ \dots \circ x_n$  for  $n \in N$   
and  $x_i \in A$

$$\{\text{jm}\}^* \ni \varepsilon, \text{ jm, jmjmjmjmjm}$$

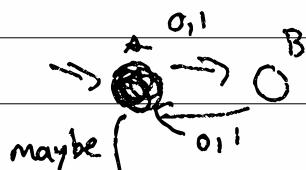
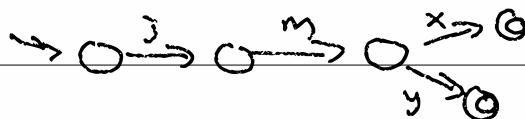
3-5/ Even Len    o Even Num



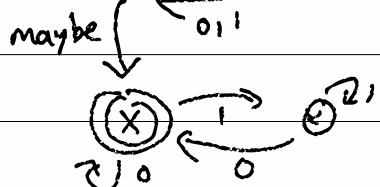
00110 ✓  
0011 x

~~00110011~~

$\{ \text{im } 3 \} \circ \{ x, y \}$

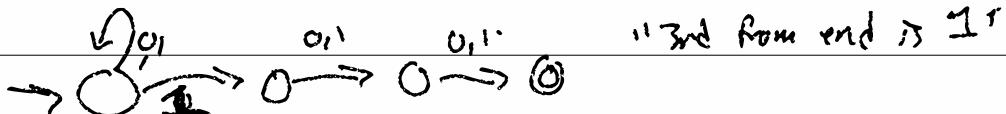


$\Sigma = \{ 0, 1 \}$



$x \in \text{DFA}$  iff

There is some path  
from  $q_0$  to  $q_f \in F$   
labelled w/  $x$



"3rd from end is 1"

3.6) NFA = non-deterministic  
finite automata

old world: the next step was obvious

$$\delta: Q \times \Sigma \rightarrow Q$$

new world: crazy options

- do you even read achar?
- which path do you take?

$$\delta': Q \times \{\text{maybe}\} \cup \Sigma \rightarrow P(Q)$$

$$\delta'(A, r) = \{A, B\}$$

$$\delta'(A, \text{maybe}) = \{C\}$$

epsilon

$$\epsilon \in \Sigma$$

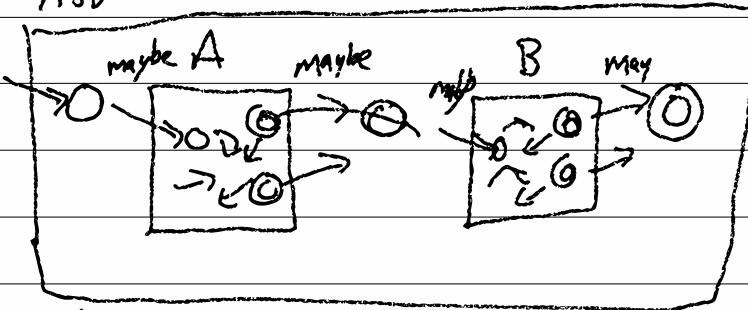
4-11  $A \circ B \in \text{DFA}$  iff  $A \in \text{DFA}$   
 $A^*$   $\wedge B \in \text{DFA}$

NFA ( $N - \underline{\text{non}} \text{ D-deterministic}$ )

$$S: Q \times (\Sigma \cup \{\text{maybe}\}) \rightarrow P(Q)$$

$$S: Q \times \Sigma \rightarrow Q$$

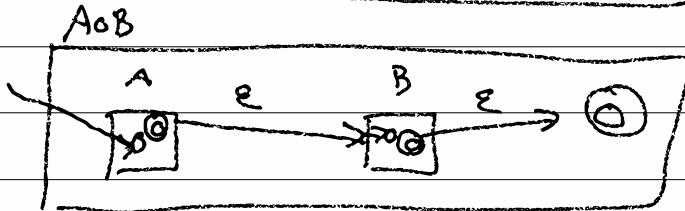
$A \circ B$



maybe written  
as "ε"

what does (NFA)

this mean?



$\text{NFA} \leftrightarrow \text{DFA}$

## 4-2] what do NFAs mean?

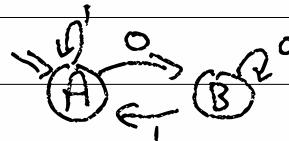
A DFA represents a set and  
a set "is" a membership function

$$U \rightarrow \{0,1\}$$

$$\subseteq \Sigma^* \rightarrow \{\text{Y}, N\}$$

$$\text{config} = \Sigma^* \times Q$$

$$\Sigma^* \rightarrow Q^*$$



$$0110 \rightarrow \underline{ABAAB} \rightarrow \text{a trace}$$

$$\Sigma^* \rightarrow (\underline{Q}, \delta)^*$$

$$0110 \rightarrow \underbrace{(0, B)(1, A)(1, A)(0, B)}_{\text{a trace}} = \Sigma^* \cup \Sigma \epsilon^3$$

$$0A1A1A0B \rightarrow N$$

$$\text{valid? } : \delta(\boxed{\Sigma}, Q)^* \rightarrow \{\text{Y}, N\}$$

$$\text{valid } g; \epsilon = Y$$

$$\text{valid } g; (c, g_j) : \text{more} = \text{if } \delta(g_j, c) = \boxed{g_j}$$

$$\text{Nvalid? } : Q \times (\boxed{\Sigma} \times Q)^* \rightarrow B$$

$$\text{valid } g; \text{ more}$$

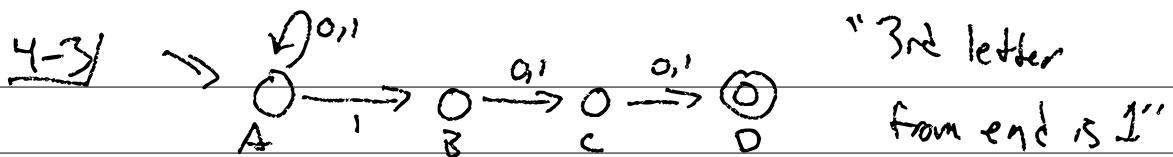
$$\text{Nvalid } g; \epsilon = Y$$

$$\text{o.w. } N$$

$$\text{Nvalid } g; (c, g_j) : \text{more} =$$

$$\text{if } \boxed{g_j} \boxed{e} \boxed{\delta(g_j, c)} \text{ then Ag Oracle}$$

$$\frac{\text{Nvalid } g; \text{ more}}{O.W. N}$$



0 1 00

1 1 1

1 1 0 1 0 0

- Y

0 0 0

1 0 0 0

1 0 1 1

- N

(0, A)(1, A)(0, A)(0, A) ✓

str( $\Sigma \times Q)^* = \Sigma^*$

(0, A)(1, B)(0, C)(0, D) ✓

str  $\epsilon = \epsilon$

(0, B)(1, C)(1, D)(0, D) X

str(c, -) : move c

$\delta(A, 0) = \Sigma A \}$

$\delta(D, 0) = \emptyset$

c o str more

accepts :  $\Sigma^* \rightarrow Y/N$

accepts  $w = Y$  iff  $\exists t \in \text{traces. } \text{str}(t) = w$ .

valid go  $t = Y$

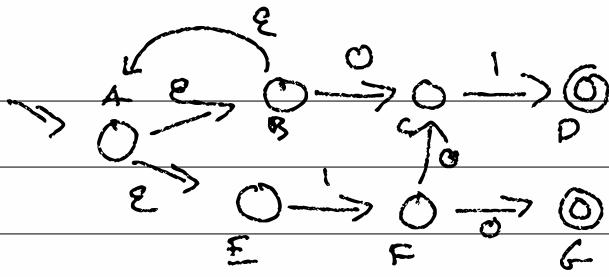
and last-state( $t$ )  $\in F$

NFA-accepts :  $\Sigma^* \rightarrow Y/N$

figure all possible traces

check if valid and if strings match

check if past is in  $\epsilon F$



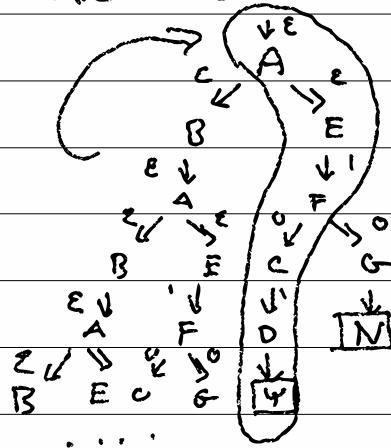
$$(\varepsilon, \beta)(0, c)(1, d) = \varepsilon \cdot 0 \cdot c \cdot 1 = 0$$

$$(\varepsilon, \Xi)(1, F)(0, C)(1, D) = 101\varepsilon_010001$$

$$(\varepsilon, E)(1, F)(0, G) \quad 10 = 20100$$

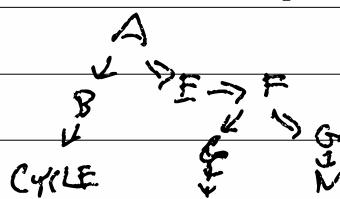
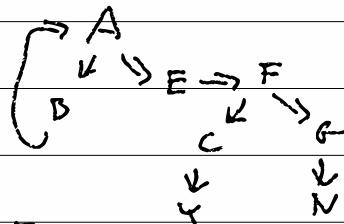
$(\epsilon, B)(\epsilon, A) \times$  where  $x$  is valid  
 $\rightarrow$  valid

Trace Tree = T | N | Branch ( $\Sigma, Q$ ) (List TT)



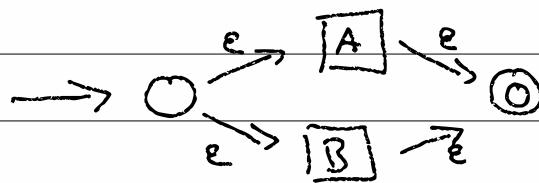
101

$\Sigma E$   $I_F$  OC 10



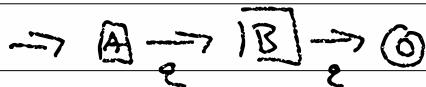
Forking model of NFAs (make TT)  
Backtracking model (explores T?)

4-5)  $A \cup B$



$x \in A$   
 $0 \rightarrow \square$   
State  $X$  transitions  
to THE start

$A \circ B$

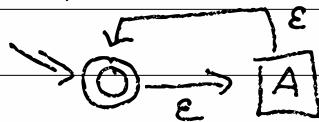


$\square \rightarrow 0$

All accepting states

of  $A$  transition to  $0$

$A^*$

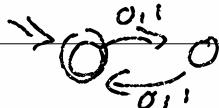


4-6)   $\forall A$ ,  $A \in \text{DFA} \Leftrightarrow A \in \text{NFA}$

$\Rightarrow$

$\Leftarrow$

DFA  $\Rightarrow$  NFA



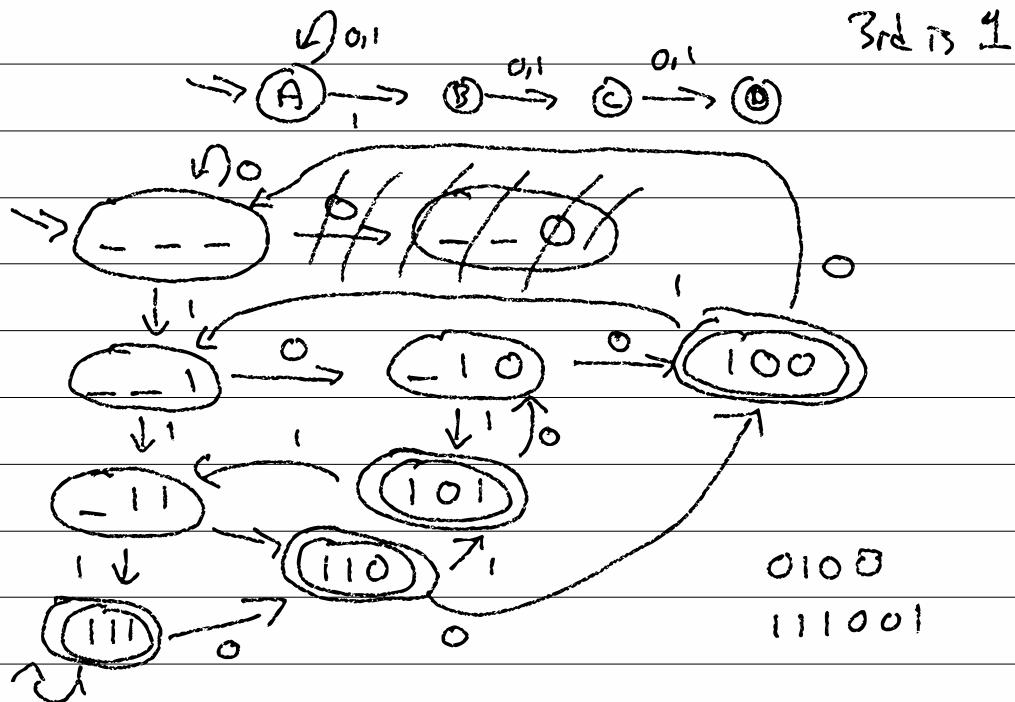
DFA  $\delta: Q \times \Sigma \rightarrow Q$

NFA  $\delta': Q \times \Sigma_c \rightarrow P(Q)$

$$\delta'(q_i, \epsilon) = \emptyset$$

$$\delta'(q_i, c \in \Sigma) = \{\delta(q_i, c)\}$$

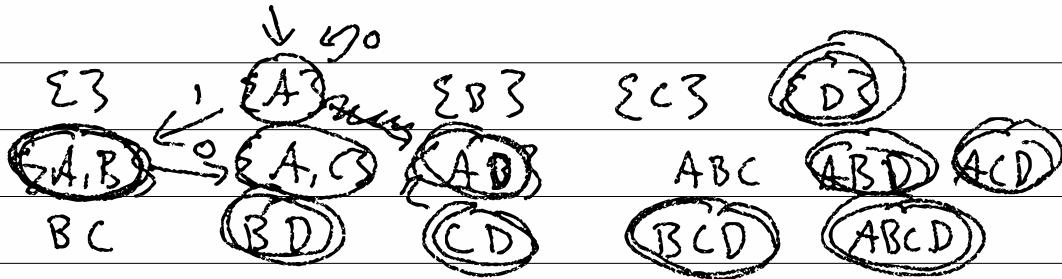
NFA  $\Rightarrow$  DFA



4-7)  $\text{NFA} = (\mathbb{Q}, \Sigma, q_0, \delta; \mathbb{P}(\mathbb{Q}) \xrightarrow{\Sigma} \mathbb{P}(\mathbb{Q}))$

$\text{DFA}^{\text{out}} = (\mathbb{Q}', \Sigma, q'_0, \delta': \mathbb{Q}' \times \Sigma \rightarrow \mathbb{Q}', F' \subseteq \mathbb{Q}')$

$$\mathbb{Q}' = \mathbb{P}(\mathbb{Q})$$



$$q'_0 = \Sigma^{q_0}$$

$F'$  = any state where  $nF \neq \emptyset$

$$\begin{aligned} \delta'(\Sigma^{q_1}, \dots, \Sigma^{q_n}, c) &= \\ \cup \quad \delta(q_i, c) \end{aligned}$$

$$\underline{5-1} / A \cup B \quad \delta_A : Q_A \times \Sigma \rightarrow Q_A$$
$$\delta_B : Q_B \times \Sigma \rightarrow Q_B$$

$$\delta' : \overbrace{Q_A \times Q_B}^{\text{(Q}_A \times \text{Q}_B)} \times \Sigma \rightarrow Q$$

$$\delta'((q_a, q_b), c) = (\delta_A(q_a, c), \delta_B(q_b, c))$$

char

$$(p, c) \Rightarrow \text{new Pair } \left( \begin{array}{l} \downarrow \\ \text{pair} < \text{State}, \text{State} \end{array} \right) \left( \begin{array}{l} \nearrow \\ \text{fst} \end{array} \right) \left( \begin{array}{l} \nearrow \\ \text{snd} \end{array} \right) \left( \begin{array}{l} \text{delta a}(p, \text{fst}, c), \\ \text{delta b}(p, \text{snd}, c) \end{array} \right);$$

## S-2/ NFA $\rightarrow$ DFA

$(Q, \Sigma, q_0 \in Q,$

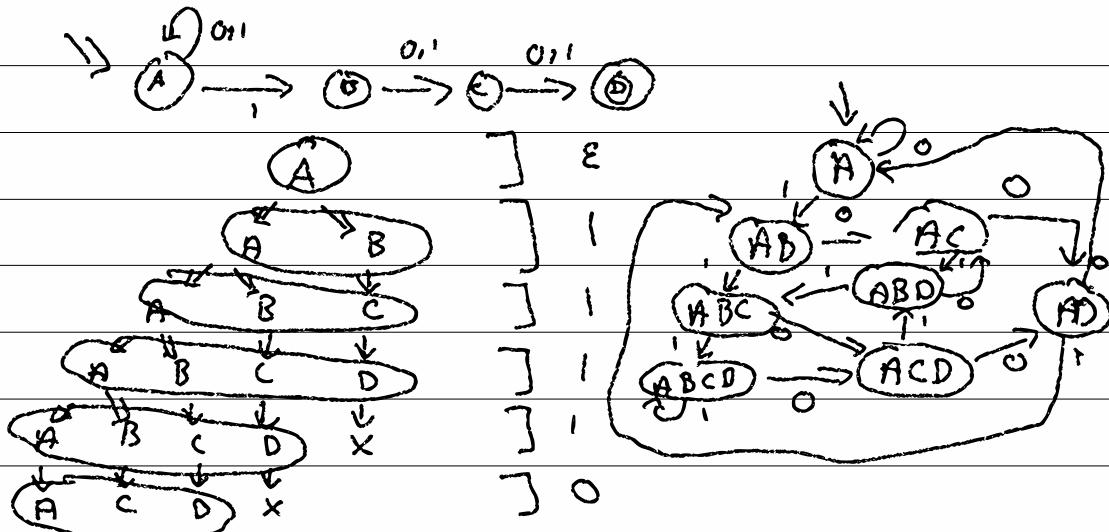
$\delta: Q \times \Sigma \rightarrow P(Q),$

$F \subseteq Q)$

$(Q', \Sigma, q'_0 \in Q'$

$\delta': Q' \times \Sigma \rightarrow Q',$

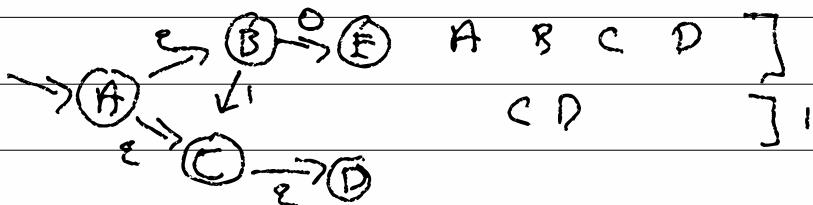
$F' \subseteq Q')$



$\begin{array}{c} \nwarrow \swarrow \\ A \quad B \\ \downarrow \quad \downarrow \\ C \end{array} ]^0$

$\begin{array}{c} A \\ \downarrow \quad \downarrow \\ A \quad B \\ ]^0 \end{array}$

$\begin{array}{cccc} A & C & C & A \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ DLAH & DLAH & DLAH & DLAH \end{array}]^x$



5-3/

$$E: Q' \rightarrow Q' \quad P(Q) \xrightarrow{P(Q)} - \text{follow all C-transitions}$$

Trace Tree DFA

$Q'$  = things at the bottom of a tree  
set = a set of states of  
the NFA =  $P(Q)$

$g_0'$  = the top of the tree  
= the set that has only the first state  
 $= E(\{\}) \in P(Q)$

$\delta'$  = maps the bottom of the tree to the next level  
= set of all next states of each state in the level of the tree

$$\delta'(Q_i, c) = \bigcup_{q_i \in Q_i} \delta(q_i, c)$$

$F$  = any level of tree with some accepting state  
= any set with an element in  $F$   
=  $\{Q_i \mid \underbrace{Q_i \subseteq Q \text{ and } Q_i \cap F \neq \emptyset}\}_{Q_i \in P(Q) = Q'}$

= (set-of-gs  $\rightarrow$   
for each  $g_i$  in set-of-gs  
if dfa.F.apply( $g_i$ ) then  
return true  
return false)

5-y)

$E(\text{set } \langle Q \rangle g_i)$

queue  $\langle Q \rangle$  next = ~~empty~~  $g_i$

set  $\langle Q \rangle$  seen = empty

while  $(\text{not } \langle Q \rangle \text{ is empty})$

$\delta(\text{next}, \text{first}, \varepsilon)$  add those

to next unless in seen

return seen

$E(A) = \text{least fixed point of}$

$E^*(A)$

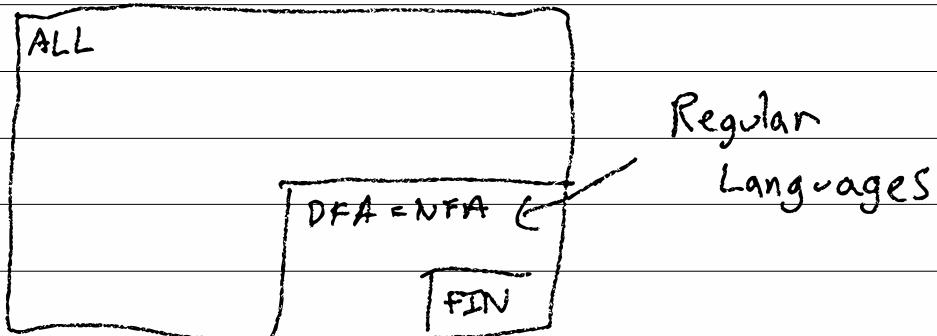
$E^*(A) = A \cup \bigcup_{g_i \in A} \delta(g_i, \varepsilon)$

6-1)  $\forall N \in \text{NFA}, \exists D \in \text{DFA}.$  compile :

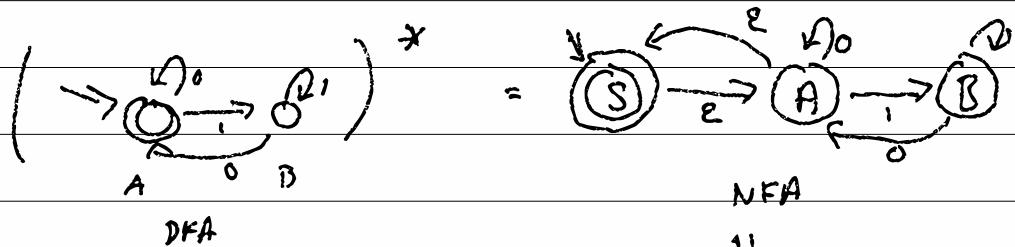
$$N = D$$

$\forall D \in \text{DFA}, \exists N \in \text{NFA}.$

$$D = N$$



Program f ...      'f; g'  
 program g ...      compositional



## 6-2 / Regular Expressions

re :=	$\epsilon$	EMPTY
	$\emptyset$	NULL
	c	Char
	re U re	CUP
	* re	STAR
	re o re	CIRC

interface RegEx { }

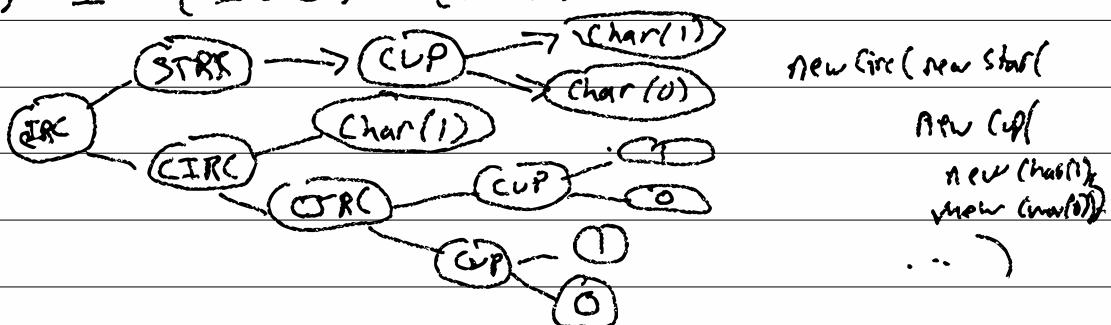
class RE\_EMPTY() impl REGEX { .. . }

RE\_NULL()

RE\_Char (char c)

RE\_Cup (REGEX lhs, REGEX rhs)

$(\text{I}\cup\text{O})^* \text{I} \circ (\text{I} \cup \text{O}) \circ (\text{I} \cup \text{O})$  - "3rd from end is I"



6-3 |  $L : RE \rightarrow \text{ALL} = P(\Sigma^*)$

fix the language if doesn't  
(or come)

$$L(\epsilon) = \{\epsilon\}$$

$$L(\emptyset) = \emptyset$$

$$L(c) = \{c\}$$

$$L(r \cup r') = L(r) \cup L(r')$$

$$L(r \circ r') = L(r) \circ L(r')$$

$$L(r^*) = L(r)^*$$

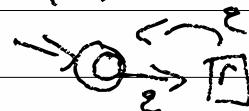
class Up {  
     $L()$   $\in$   
    return up(lhs, L(), rhs, L());  
}

compile : RE  $\rightarrow$  NFA

$$\text{compile } (\epsilon) = \xrightarrow{\epsilon} \textcircled{0}$$

$$\text{compile } (r^*) =$$

$$\text{compile } (\emptyset) = \xrightarrow{\epsilon} \textcircled{0}$$



$$\text{compile } (c) = \xrightarrow{\epsilon} \textcircled{0} \xrightarrow{c} \textcircled{0}$$

$$\text{compile } (r \cup r') = \xrightarrow{\epsilon} \textcircled{0} \xrightarrow{c} \boxed{r} \xrightarrow{\epsilon} \boxed{r'} \xrightarrow{\epsilon} \textcircled{0}$$

$$\text{compile } (r \circ r') = \xrightarrow{\epsilon} \boxed{r} \xrightarrow{\epsilon} \boxed{r'} \xrightarrow{\epsilon} \textcircled{0}$$

6-4

$$"\ " = \epsilon$$

$$\underline{\underline{\quad}} = \emptyset$$

$$"\ c" = c$$

$$"\ xyz^*" = x \circ y \circ (z^*)$$

$$"\ (xyz)^*" = (x \circ y \circ z)^*$$

$$"\ xyz" = x \circ y \circ z$$

$$"\ [abc]" = (a \cup b \cup c) \quad (a, b, c \in \Sigma)$$

$$"\ (a \mid b \mid c)" = \Rightarrow \quad (a, b, c, \in \Sigma^*)$$

[012]

(zero | one | two)

$$\cdot = \epsilon$$

$$"\ .^* \backslash. m; s" = \Sigma^* \circ ' . ' \circ ' m' \circ ' ; ' \circ ' s'$$

8-5/

gen : RE  $\rightarrow \Sigma^*$  or false

gen  $\epsilon = \epsilon$

gen  $\emptyset = \text{FALSE}$

gen  $c = 'c'$  flip coin

gen  $x \cup y = \text{gen } x \sqcup \text{gen } y$

gen  $x \circ y = \text{gen } x \circ \text{gen } y$

gen  $x^* = \boxed{\square}$

= gen  $(\epsilon \cup x \circ x^*)$   
"mjs"

equal : RE  $\times$  RE  $\rightarrow$  Bool

equal  $x \ y =$

NFA2DFA(compile  $x$ )  $\xrightarrow{\text{?}} \text{def equality?}$   
NFA2DFA(compile  $y$ )

$$(\bar{A} \cap B) \cup (A \cap \bar{B}) = \emptyset$$

G-6)

$$x + 0 = x$$

$$x = x$$

$$x \cdot 1 = x$$

$$\frac{a = b}{a+x = b+x}$$

$$\frac{a = b}{ax = bx} \quad x \neq 0$$

$$2(3x + 17) = 6x + 34 \quad \text{"algebra"}$$

$$3x + 17 = 3x + 17$$

$$3x = 3x$$

$$x = x$$

$$17 \cancel{x} \approx 17 \cancel{z}?$$

WZSS

$$\emptyset \cup x = x \cup \emptyset = x$$

$$\emptyset \circ x = x \circ \emptyset = \emptyset$$

$$\varepsilon \circ x = x \circ \varepsilon = x$$

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

$$\approx \varepsilon \cup \emptyset = \varepsilon$$

$$x \circ (y \cup z) = x \circ y \cup x \circ z$$

NFA  $\approx$  DFA      in:  $N$  states      ( $Q$ )

out:  $z^N$  states      ( $P/Q$ )

6-7/ size : RE  $\Rightarrow$  Nat

$$\text{size } \emptyset = 1$$

$$\text{size } \varepsilon = 1$$

$$\text{size } c = 1$$

$$\text{size } (x \cup y) = \text{sz } x + \text{sz } y + 2$$

$$\text{size } (x \circ y) = \text{sz } x + \text{sz } y + 1$$

$$\text{size } (x^*) = \text{sz } x + 1$$

$$x \circ (y \cup z) = x \circ y \cup x \circ z$$



emacs \* .c

↓ bash

emacs .c

↓  $\Sigma^* \circ \omega C$

compile

→ o → o → o →

+

→ o → o → o

↓ N2D

Yf S

304

← ↗ ↘

304.c main.c

dir

304.c

awesome-quotes.txt

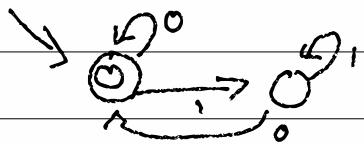
mycat.jpg

main.c

+

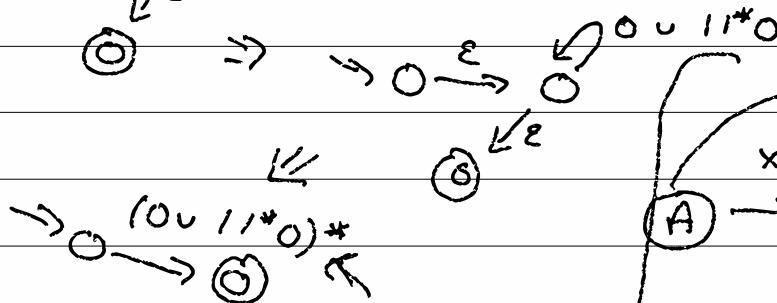
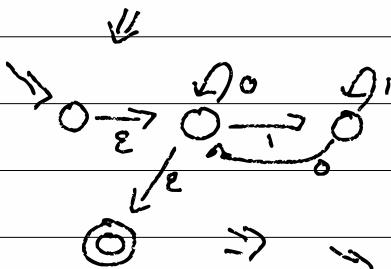
emacs

6-8] decompire : DFA  $\xrightarrow{(\text{NFA}) \text{ or }} \text{RE}$



$$\Sigma^* \circ 0 \cup \epsilon$$

$$(1 \cup 0)^* \circ 0 \cup \epsilon$$

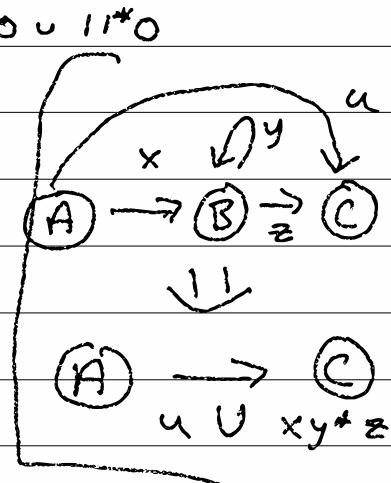


$$(0 \cup 11^* 0)^*$$

$\epsilon \quad 0 \quad 10 \quad 111110$

$011101111100$

11100011101100101010



G-9 / decompile : N-state NFA  
→ RE

START : N-NFA → (N+2)-GNFA

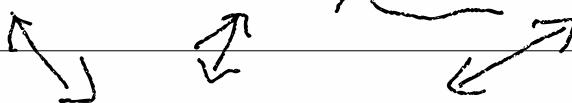
RIP : (N+1)-GNFA → N-GNFA

END : 2-GNFA → RE

decompile m = end ∘ rip<sup>n</sup> ∘ start (n)

7-1) DFA/NFAs  $\rightarrow$  RE

DFA<sub>s</sub>  $\leftrightarrow$  NFA<sub>s</sub>  $\leftrightarrow$  RE



Regular  
Languages

NFA  $\rightarrow$  RE

IN: n-NFA  $\rightarrow$  (n+2)-GNFA

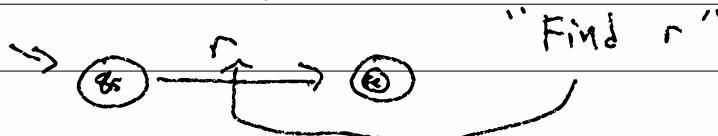
RIP<sup>n</sup>: (n+1)-GNFA  $\Rightarrow$  n-GNFA

OUT: 2-GNFA  $\rightarrow$  RE

GNFA =  $(Q, \Sigma, g_s, g_e, \Delta : (Q-g_e) \times (Q-g_s) \xrightarrow{e_Q} e_Q \rightarrow RE(\Sigma))$   
 $S: Q \times \Sigma \rightarrow Q$

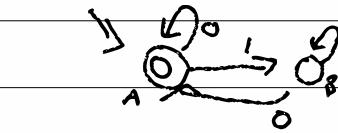
OUT: 2-GNFA  $\rightarrow$  RE

$(\{\Sigma, g_s, g_e\}, \Sigma, g_s, g_e, \{(g_s, g_e), \uparrow\})$   
 $= r = \Delta(g_s, g_e)$



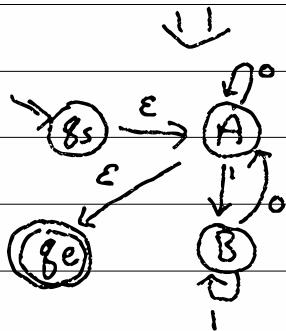
7-2) IN: NFA  $\Rightarrow$  GNFA (n+2) -  
 $(Q, \Sigma, g_0, \delta: Q \times \Sigma \rightarrow P(Q))$   $(Q', \Sigma, g_s, g_e,$   
 $F, \Delta: (Q' - g_e) \times (\Sigma - \{g_e\}) \rightarrow R_E)$

$$Q' = Q \cup \{g_e, g_s\}$$



$$\Delta(g_i, g_j) = r$$

$$\Delta(g_s, g_0) = \varepsilon$$



$$\Delta(g_s, g_j \neq g_0) = \emptyset$$

$$\Delta(g_f \in F, g_e) = \varepsilon$$

$$\Delta(g_i \notin F, g_e) = \emptyset$$

$$\Delta(g_i, g_j) = \cup \{\varepsilon_{g_{ij}}\}$$

$$\delta(g_i, c) \ni g_j \}$$

7-3/ RIP =  $(n+1)$ -GNFA  $\rightarrow$   $n$ -GNFA  
 main  $'(Q, \Sigma, g_s, g_e, \Delta)'$   $\downarrow$   $'(Q', \Sigma, g_s, g_e, \Delta')$

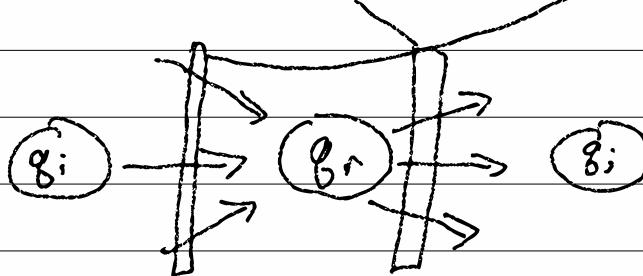
↓  
 jay  
 main

$$Q = Q' \cup \{ q \text{ gonna be killed} \}$$

↓  
 jay  
 main  
 ↓  
 exit  
 ↓  
 exit

$$\Delta' : \underbrace{(Q' - g_e)}_{g_r \in} \times \underbrace{(Q' - g_s)}_{g_r \in} \rightarrow \text{RE}$$

$$\Delta : \underbrace{(Q - g_e)}_{g_r \in} \times \underbrace{(Q - g_s)}_{g_r \in} \rightarrow \text{RE}$$

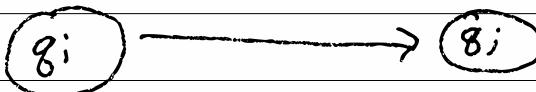


$$x \circ y^* \circ z \cup \alpha$$

$$\rightarrow x \circ z$$

$$\Delta'(q_i, q_j)$$

$$=$$

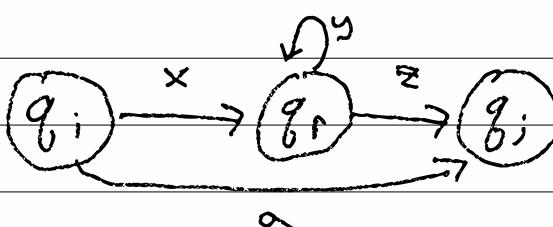


$$x = \Delta(q_i, q_r)$$

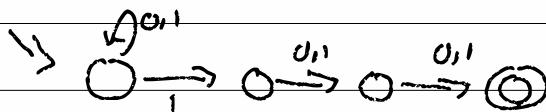
$$y = \Delta(q_r, q_r)^*$$

$$z = \Delta(q_r, q_j)$$

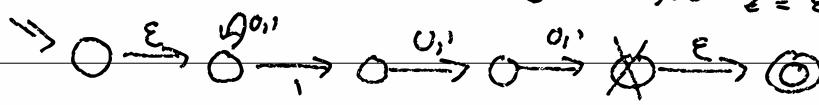
$$\alpha = \Delta(q_i, q_j)$$



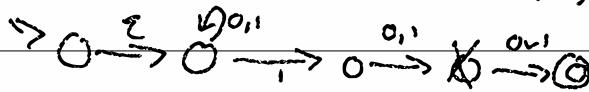
~~7-4/ int main () {~~  
 int ~~x = 8;~~  
~~int y = f(x, z);~~  
 ...  
 int y = x + 2 \* 2;  
 ↓  
 int f(int ~~z~~){  
 int ~~z~~;  
 return ~~z + 2 \* 8;}~~



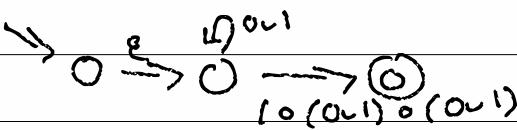
$$\text{out} = x \quad y = d \quad z = e \quad a = \emptyset \quad \Downarrow \text{IN}$$



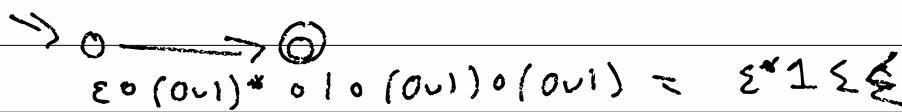
$$x \circ y \circ z \circ u \circ v = (x \circ 1) \circ 0 \circ 0 \circ 0 \circ u \circ v \\ = (x \circ 1)$$



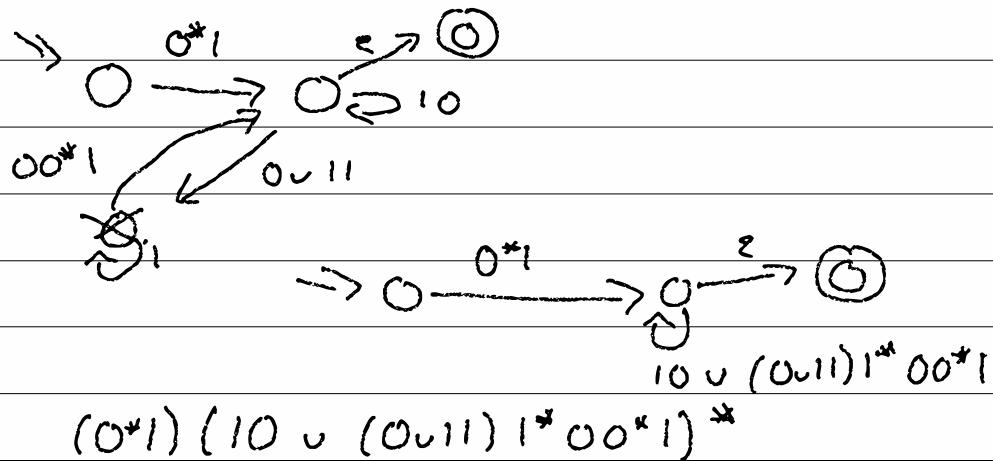
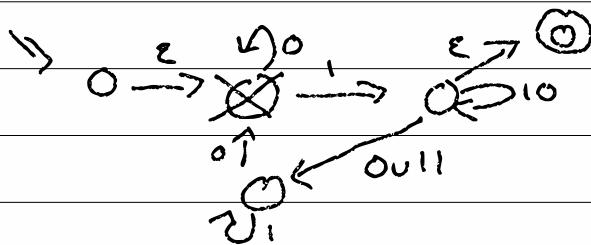
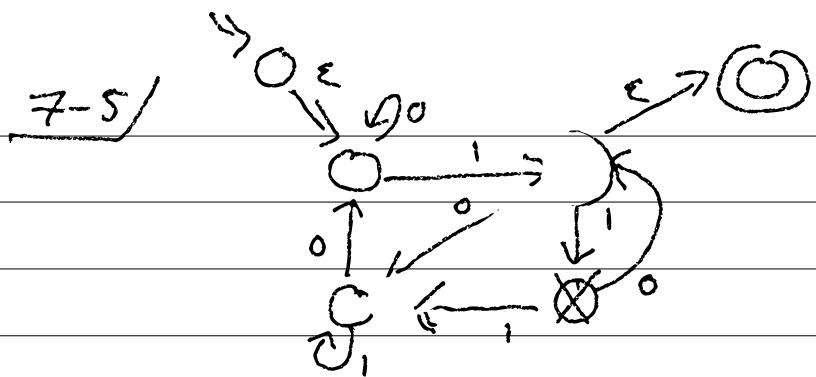
$$(out) \circ (out)$$



$$1 \circ (out) \circ (out)$$

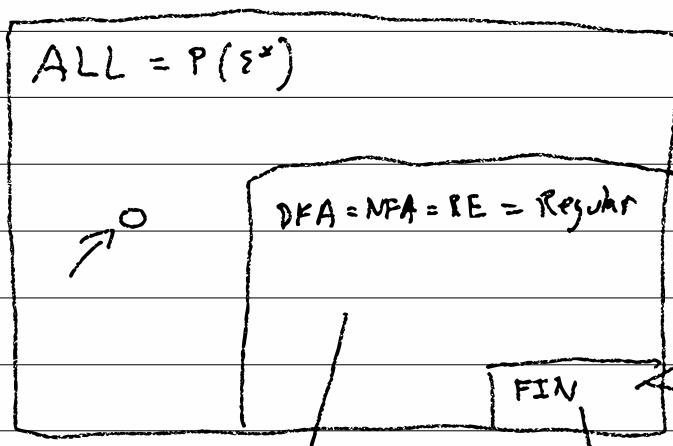
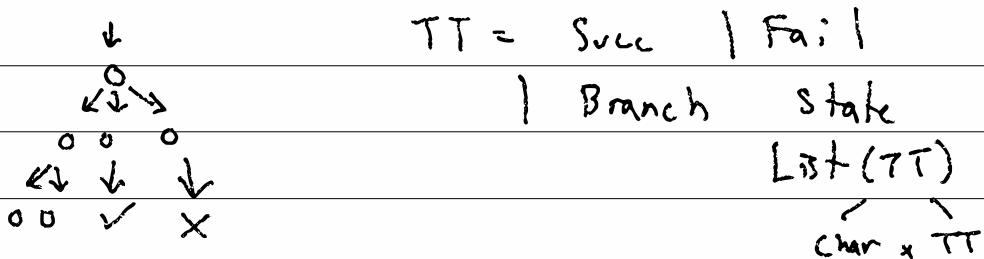


$$\varepsilon \circ (out)^* \circ 1 \circ (out) \circ (out) = \varepsilon^* 1 \subseteq \Sigma^*$$



8-1 ε Object → Char (char c)  
 Upsilon () Epsilon ()  
 UTF-8

TT



$\{\text{all strings ending in } 0\}$

$\{\epsilon, 00\}$

$$A \in REG \iff \exists d \text{ DFA}, L(d) = A$$

$$\neg A \in REG \iff \neg (\exists d) \iff \forall d, d \text{ DFA} \wedge$$

$$\neg \exists x, P(x) \iff \forall x, \neg P(x) \quad L(d) \neq A$$

$$\neg \forall x, P(x) \iff \exists x, P(x)$$

8-2/ How can we know stuff about  
infinite sets?

$\forall x \in A, P(x)$

$P: DFA \rightarrow \text{Prop}$

$\Rightarrow z_0 \in Q$

$P: IP(\Sigma^*) \rightarrow \text{Prop}$

$\neg P(B)$  (where  $B \in IP(\Sigma^*)$  and we "hope"  
 $B$  isn't in DFA)

Q. What is P?

1. Prove  $\nexists RFG, P(A)$

Pumping  
lemma

2. Prove  $\exists B \in \text{RELL}, \neg P(B)$

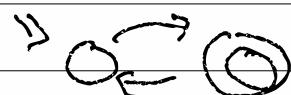
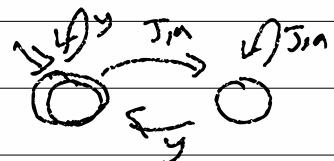
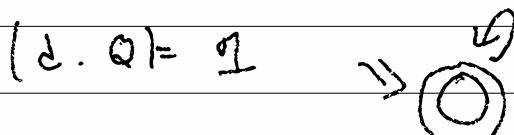
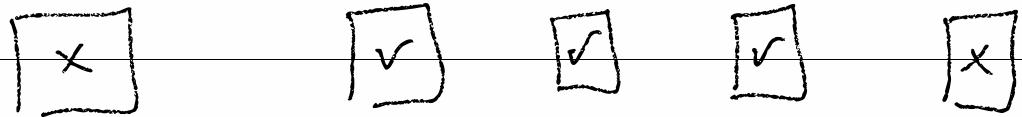
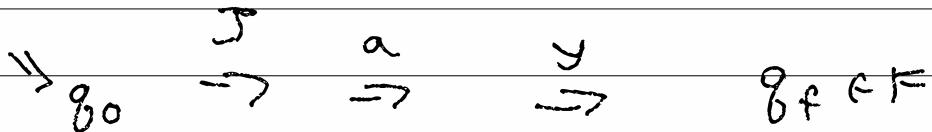
conclude  $B \notin \text{REG}$ .

$$\Sigma = \dots$$

8-3)

"Jay" & d

$\Sigma \ni \{s, a, y\}$



Daphne wins

pick a number of states ; 4

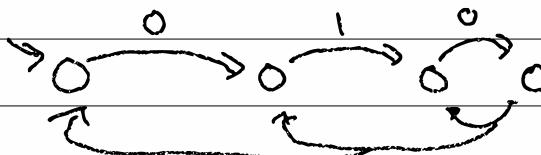
she picks a char

I say what state we goto

I win if I never say same state

she wins if I repeat

How many turns to win?

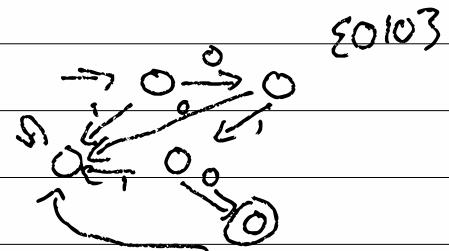
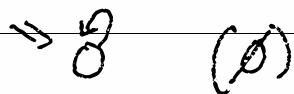
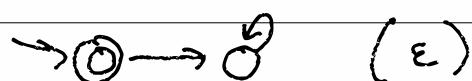


$4 \Rightarrow N \Rightarrow$

|@|

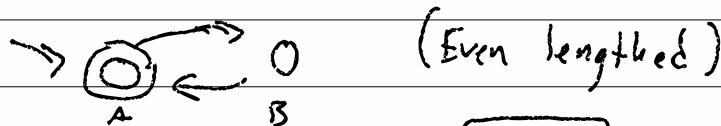
$\delta: Q \times \Sigma \rightarrow Q$  Total Fun  
 $\xrightarrow{\text{from}} \xrightarrow{\text{to}}$

All DFAs have a loop



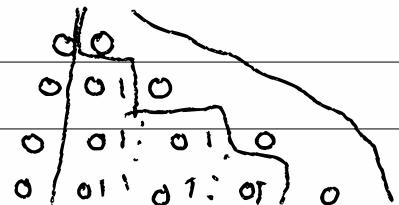
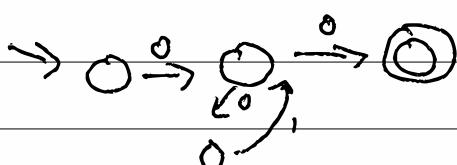
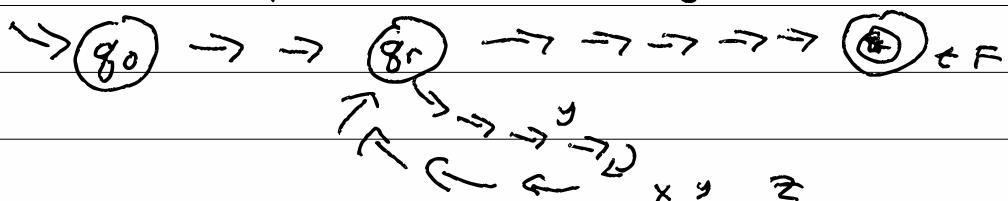
Some have "exciting" loops

$\exists s \in (\text{DFA})$ ,  $s = \dots \dots \dots \dots \dots$



$$\begin{aligned} \epsilon &= A \\ &= \epsilon \circ \epsilon \circ \epsilon \end{aligned}$$

$$0110 = \overbrace{ABA\bar{A}}^z = \epsilon \circ 01010$$



8-5/ If a machine hasn't an existing loop ...

They are all finite  
 $L(m) \in FIN$

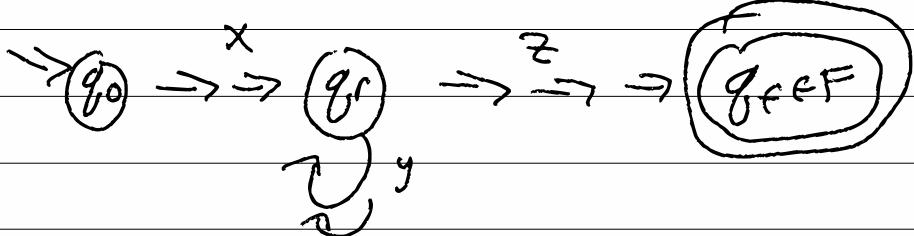
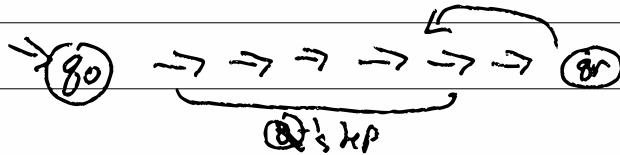
If it does have one ...

then it must be infinite (regular)

If there is an existing loop ...

what is the shortest string to  
"find" ;+ m?

$$|s| = |Q| \quad (s \in L(\delta))$$



$\in \text{ALL}(\text{a language}) (\mathcal{P}(\Sigma^*))$

8-6/ RPP(A) = Regular Pumping Property

? ( $\exists p \in N, \quad / / \quad p = |Q|$

$\forall s \in A \quad | \quad |s| \geq p \quad )$

$\exists (x, y, z \in \Sigma^*) \quad | \quad s = xyz \quad \wedge$

$|xy| \leq p$

$|y| > 0 \quad )$

$\forall i \in N,$

$x \circ y^i \circ z \in A$

)

$\forall d \in \text{DFA}, \exists n \in \mathbb{N}, L(d) = L(n) \quad - \text{Cof}$

$\neg \text{RPP}(B) :=$

$\forall p \in N,$

$\exists (s \in B \quad | \quad |s| \geq p \quad )$

$\forall (x, y, z \in \Sigma^*) \quad | \quad s = xyz \wedge |xy| \leq p \wedge |y| > 0 \quad \text{and} \quad$

$\exists i \in N,$

$x \circ y^i \circ z \notin B$

8-7) Need: an infinite space problem

```
O* 1 { while (getc() == '0') {  
    ungetc()  
    if (getc() == '1') { net false }  
    return getc() == EOF;  
} = 232 vint2_+
```

$\forall n \in \mathbb{N}$ ,

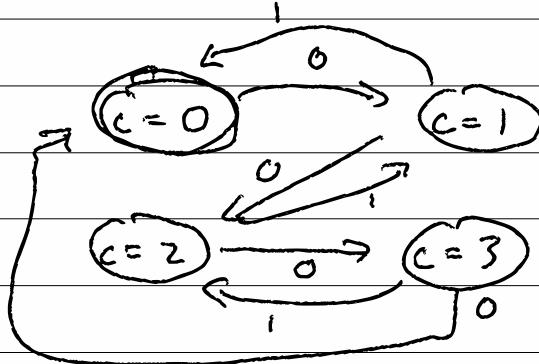
$0^n 1^n \in B$

vint count = 0;

while (in == 0) count++

while (in == 1) count--  
if (in == EOF) < count--  
return count == 0 ;

{ net false } }



q-1) RPP(A)  $\rightarrow$  Language =  $P(\Sigma^*) \approx A\bar{A}$

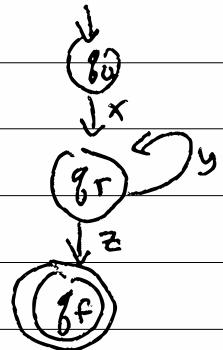
$$\exists p \in N, \quad |P| = |Q|$$

$$\forall s \in A \mid |s| > p \}.$$

$$\exists (x, y, z \in \Sigma^*) \mid s = xyz$$

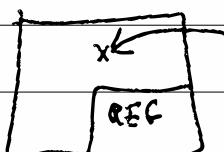
$$\forall i \in N, \quad |xy| < p$$

$$xyi z \in A \quad |y| > 0$$



$\rightarrow \text{RPP}(A) =$

ALL



$$\forall p \in N,$$

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

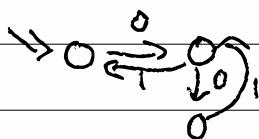
$$\forall (x, y, z \in \Sigma^*) \mid s = xyz \wedge |y| < p$$

$$\exists i \in N$$

$$xyi z \in A$$

$0^n 1^n$  ie  $x \in 0^n 1^n$

iff  $\exists n \in N, x = 00 \dots 01 \dots 1$

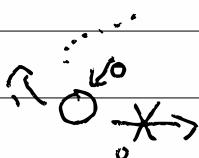


int count = 0

while (see 0) c++

while (see 1) c--

return count = 0



9-2)  $\neg \text{RPP}(\mathcal{O}^n, \mathcal{O}^n) =$

$\forall p \in \mathbb{N},$

$\exists (s \in \mathcal{O}^n)^n \mid |s| \geq p)$

choose  $s: s = x = z$

$s = \mathcal{O}^{p/2} \cup \mathcal{O}^{p/2}$

$s = \mathcal{O}^p, |s| = 2p \geq p$

$\forall (x, y, z \in \Sigma^*) \mid s = xyz \text{ and}$

$|xy| \leq p \text{ and}$

$|y| > 0$ )

$x = \mathcal{O}^a \quad a+b+c = p \quad a+b < p$

$y = \mathcal{O}^b \quad d = p \quad b > 0$

$z = \mathcal{O}^c \cup d$

$\exists i \in \mathbb{N}, xyiz \in \mathcal{O}^n, \mathcal{O}^n$

$xyiz = \mathcal{O}^a \mathcal{O}^b \cup \mathcal{O}^c \cup d = \mathcal{O}^{a+b+c} \cup d$

$a+b+c = d$

$\frac{b(i-1)}{b} = \frac{-p}{b} \quad i=0 \quad \boxed{i=1}$

fun  $n:$

if  $n = 0 : \text{ret } p \circ$

$p \circ : \mathbb{P} \circ$

else: let  $m = n-1$

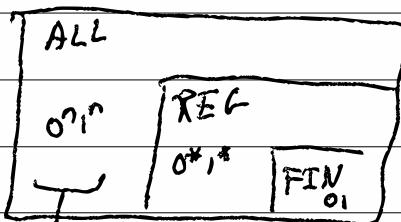
$p \circ : \text{th. } p_n \rightarrow p_{(n+1)}$

let  $p_m = \text{rec } m \Rightarrow \forall n: p_n$

$p \circ p_m$

induction on  $N:$

9-3) There is stuff outside REG'



infinitely big

All computers  
are DFAs.

if  $o^n1^n \in \text{REG}$

~~then~~<sup>and</sup>  $B \in \text{REG}$

then  $o^n1^n \circ B \in \text{REG}$

$\Sigma_1 \quad \Sigma_0 \quad \text{CFL}$

ALL [ ] | ] REG