

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
 $\forall x, x \notin \emptyset$

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

$\{\checkmark, \square\}$

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

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

union - \cup

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

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

[=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 , $\{\}$, \cup are finite.

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

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 = Σ^* ← 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 { }

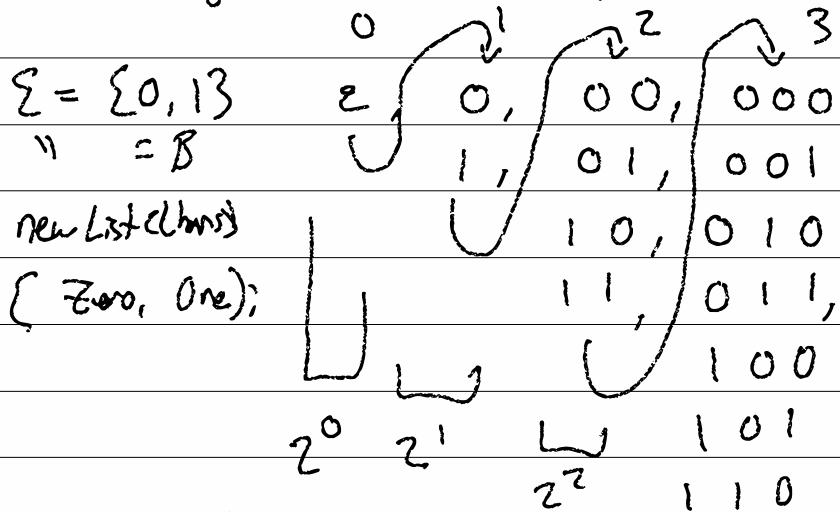
OneString (char c, String s) { ... }

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

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

(EP)

I-6/ Every alphabet has a lexicographical ordering of the strings in Σ^*



$|A|^i$ where i is layer 2^i
looking at $\underbrace{1 \ 1 \ 1}_{2^3}$

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

lexi $B \ 0 = \emptyset$

lexi $B \ 1 = 0$

lexi $B \ 2 = 1$

lexi $B \ 6 = 101$

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 Σ Universe Σ^*

{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

- ALL
- True math
 - G-TMs
 - Even strings

$\overline{\text{FIN}}$

$\emptyset \in \text{FIN}$

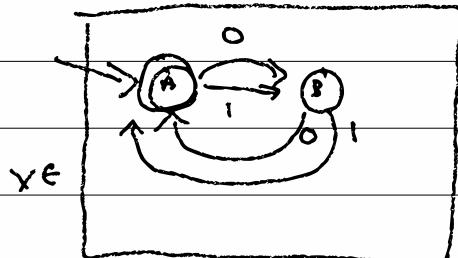
$\forall x \in \Sigma^*, \{x\} \in \text{FIN}$

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

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

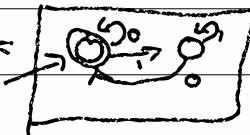
All even strings =

DFA - a deterministic finite automata



Σ or Σ^*

even numbers =



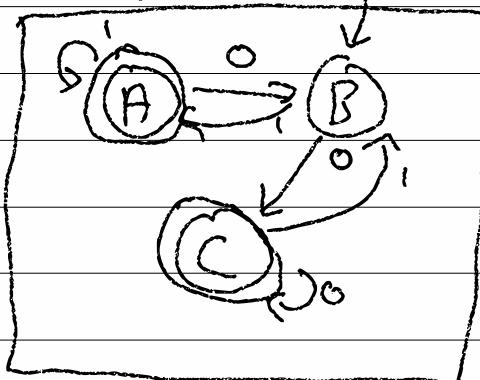
○ - states $\Sigma A, B \}$

→ ○ - start state A

○ - accepting states $\Sigma A \}$

$\overset{x}{\rightarrow} \circ$ - transition

x - labels are Σ



$\Sigma A, B, C \}$

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

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 Σ

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

public bool accepts (String x) {

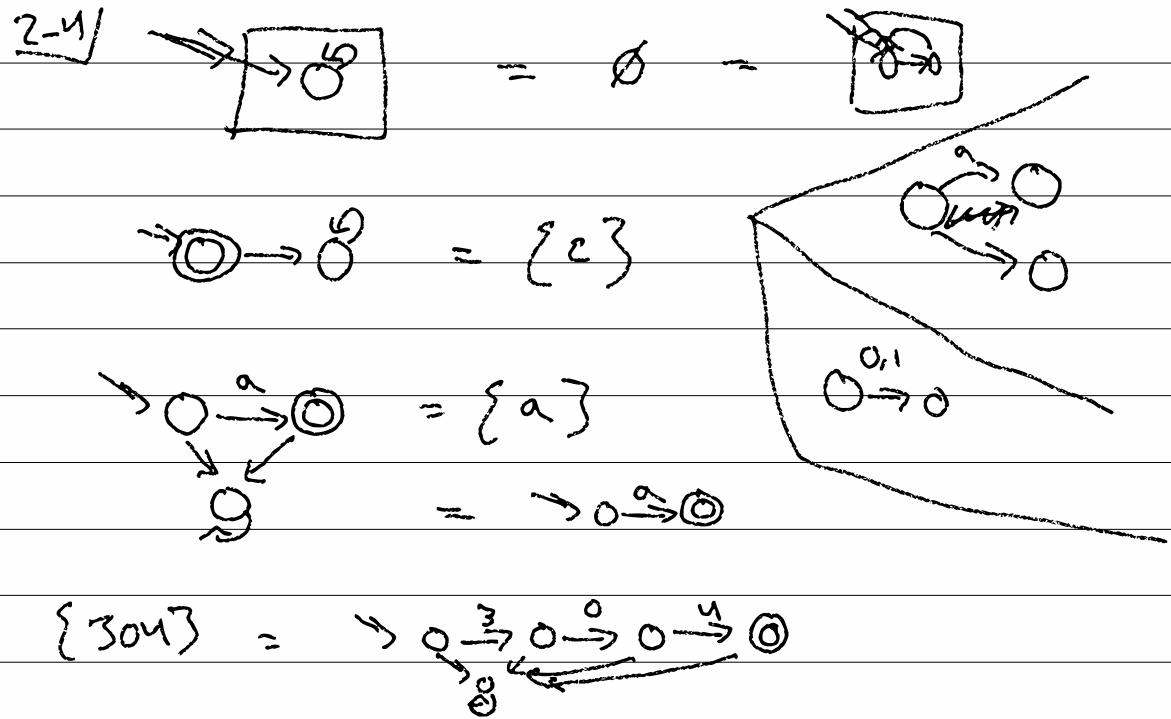
State $q_i = q_0;$

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

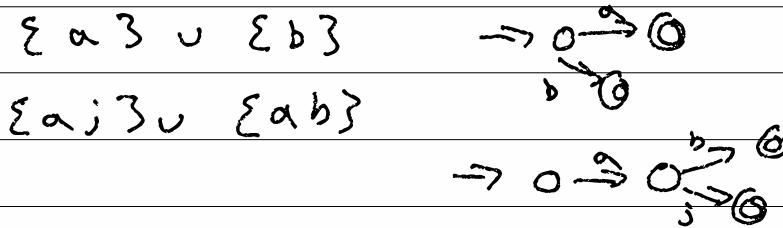
$q_i = \delta(q_i, x, \text{first}());$

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

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



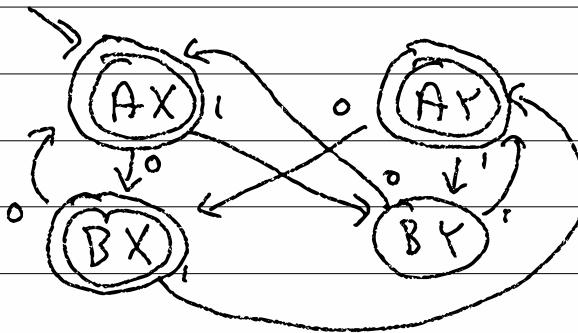
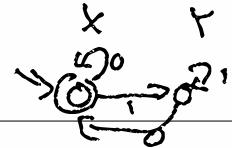
$A \cup B \leftarrow \text{DFA}$ (if $A \in \text{DFA}$ and $B \in \text{DFA}$)



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:

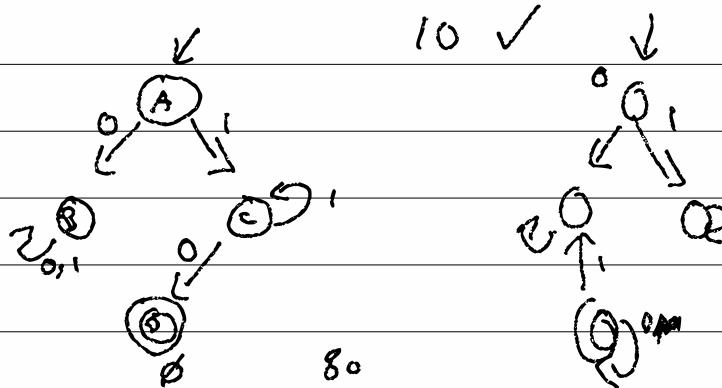
Q : Sstate \Rightarrow Bool

Σ : 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.'

$\Sigma = \{D\}$

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

$[D]$

$D \Rightarrow C, 0$

$\Sigma = \{E\}$

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

$[E]$

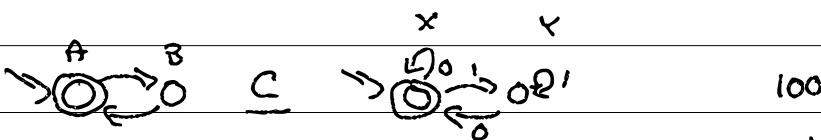
or, if not, No!

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!



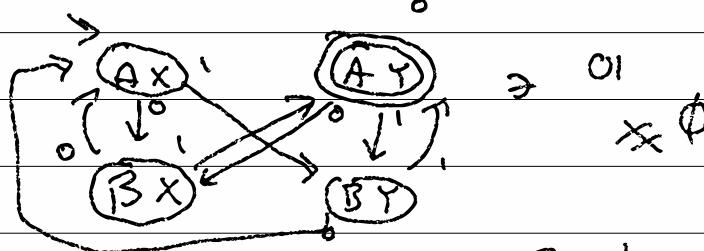
$$\boxed{\bar{A}} \subseteq \boxed{\bar{B}} \cap \boxed{\bar{A}} = \boxed{\bar{A}}$$

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

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
 $\text{model} = \text{theory}$

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

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

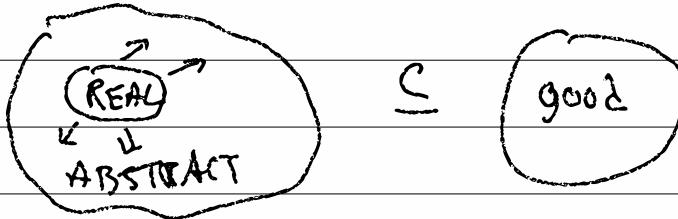
if $x > 0$ then

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

o.w.

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

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



<u>3-w)</u>	Finite	=	\emptyset	Σ^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
 $x \in A \wedge y \in B$

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

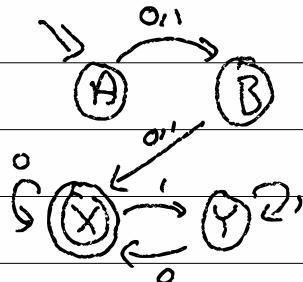
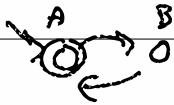
$$abcd = ab \circ cd$$

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

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

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

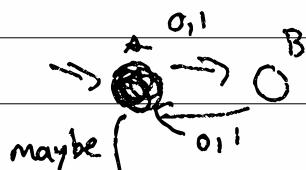
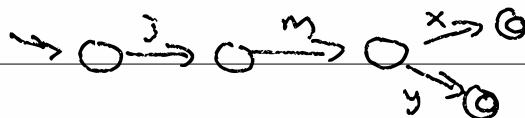
3-5/ Even Len o Even Num



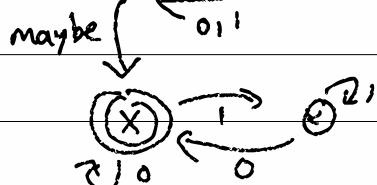
00110 ✓
0011 x

~~00110011~~

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

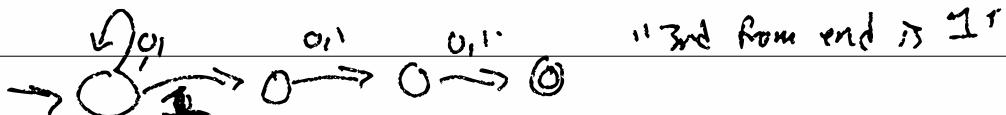


$\Sigma = \{ 0, 1 \}$



$x \in \text{DFA}$ 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^* $\wedge B \in \text{DFA}$

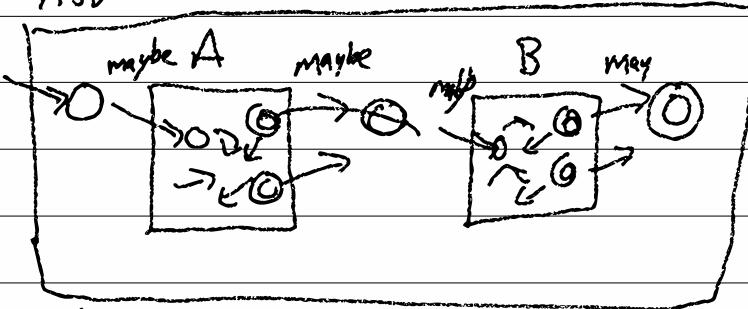
NFA ($N - \underline{\text{non}} \text{ D-deterministic}$)

$$\downarrow \qquad \qquad \qquad \downarrow$$

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

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

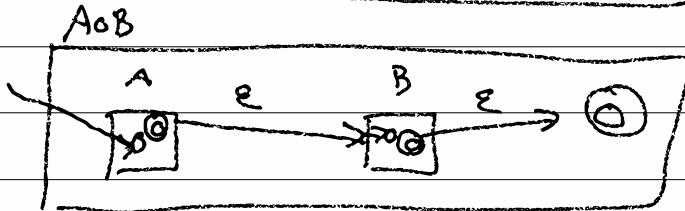
$A \circ B$



maybe written
as "ε"

what does (NFA)

this mean?



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

4-2] what do NFAs mean?

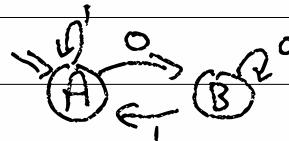
A DFA represents a set and
a set "is" a membership function

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

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

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

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



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

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

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

$$0A1A1A0B \rightarrow N$$

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

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

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

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

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

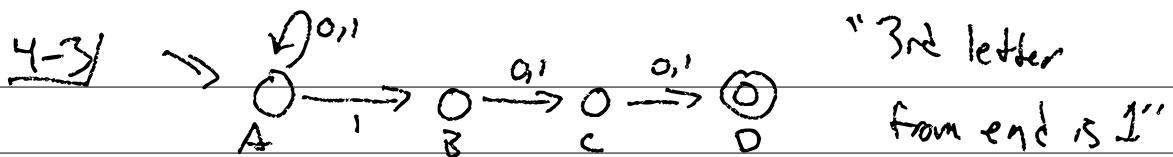
$$\text{Nvalid } g; \epsilon = Y$$

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

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

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

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



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 $(\Sigma \times Q)^*$ = Σ^*

(0, A)(1, B)(0, C)(0, D) ✓

str $\epsilon = \epsilon$

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

str (C, \cdot) : move \in

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

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

C o str more

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

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

valid $\Leftrightarrow t \in F$

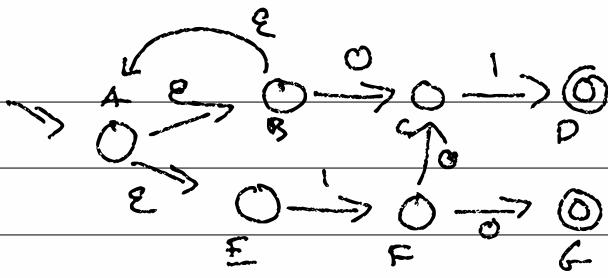
and last-state(t) $\in F$

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

figure all possible traces

check if valid and if strings match

check if past is in ϵF



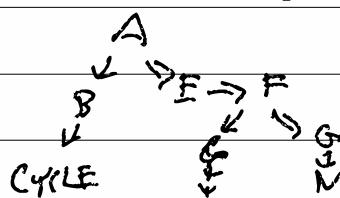
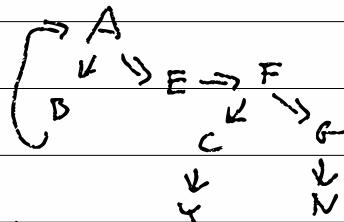
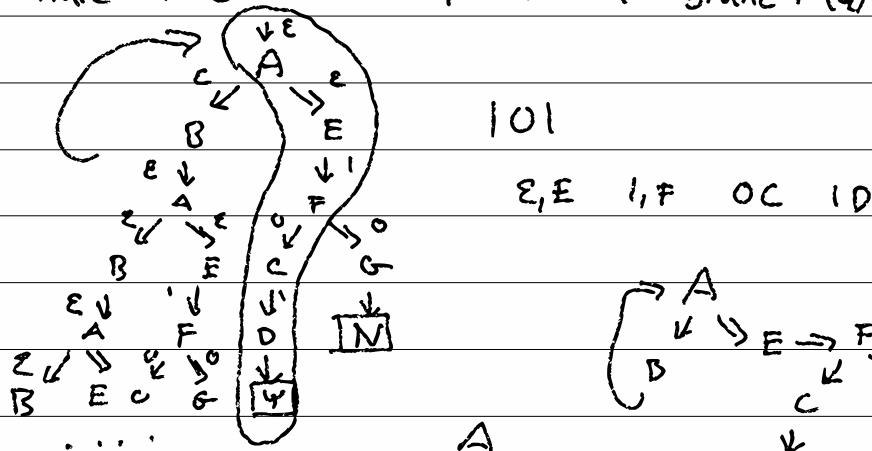
$(\epsilon, B)(0, C)(1, D) \quad 01 = 0001$

$(\epsilon, E)(1, F)(0, G) \quad 101 = 01001$

$(\epsilon, E)(1, F)(0, G) \quad 10 = 0100$

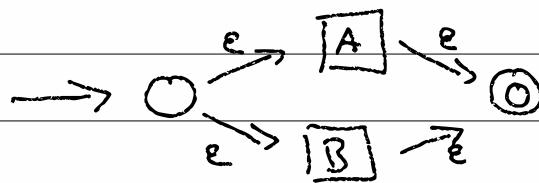
$(\epsilon, B)(\epsilon, A) \times \text{ where } \times \text{ is valid}$
 $\rightarrow \text{valid}$

Trace Tree = T | N | Branch (ϵ, Q) (List TTI)



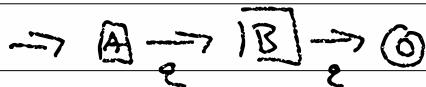
Forking model of NFAs (make TT)
 Backtracking model (explores TT)

4-5) $A \cup B$



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

$A \circ B$

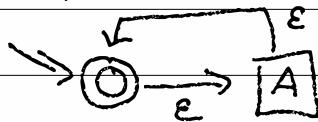


$\square \rightarrow \square$

All accepting states

of A transition to \square

A^*

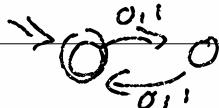


4-6) $\forall A$, $A \in \text{DFA} \Leftrightarrow A \in \text{NFA}$

\Rightarrow

\Leftarrow

DFA \Rightarrow NFA



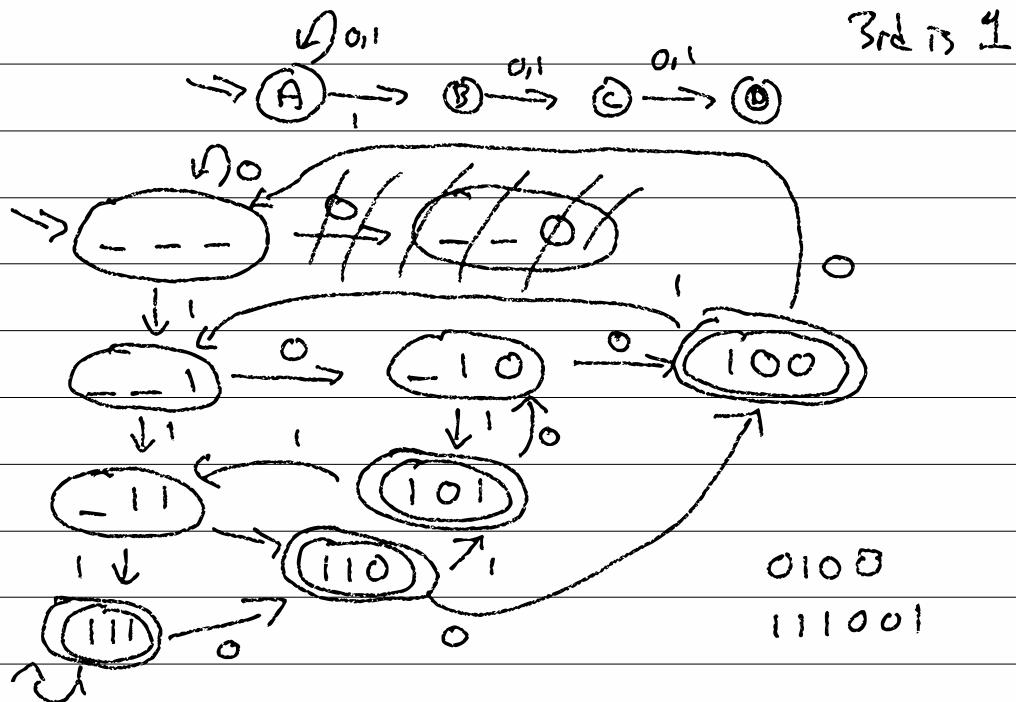
DFA $\delta: Q \times \Sigma \rightarrow Q$

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

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

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

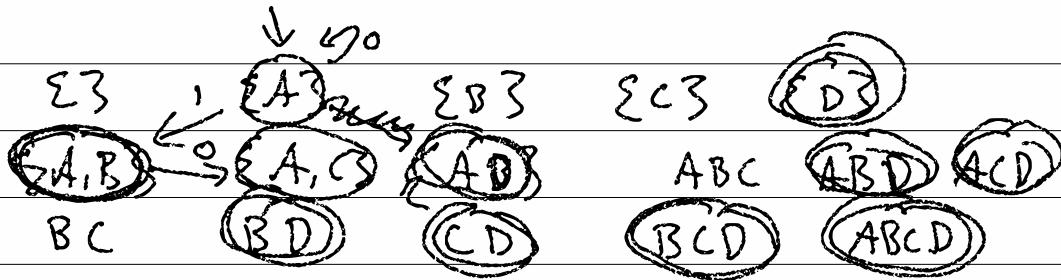
NFA \Rightarrow DFA



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

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

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



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

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

$$\delta'(\{q_1, \dots, q_n\}, c) = \bigcup \delta(q_i, c)$$

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

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

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

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

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

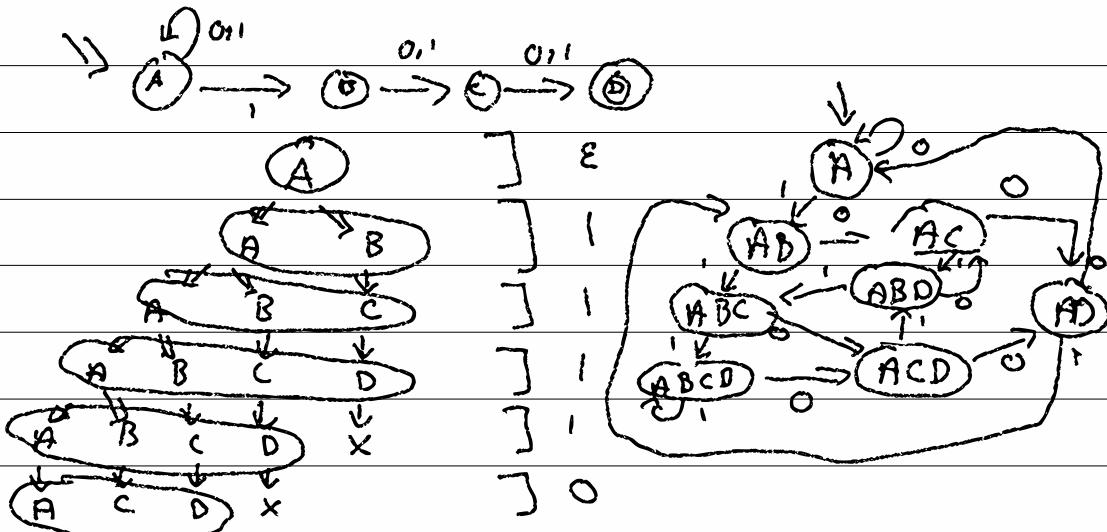
$\delta: Q \times \Sigma \rightarrow P(Q),$

$F \subseteq Q)$

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

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

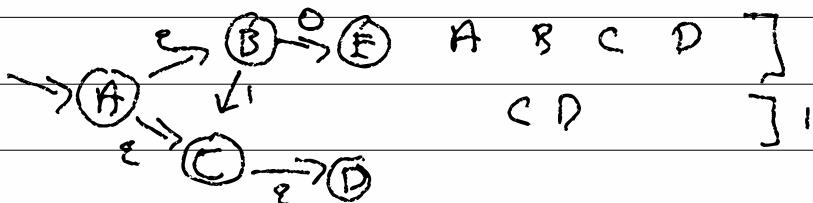
$F' \subseteq Q')$



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

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

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



5-3/

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

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
 $= E(\{\text{go}\}) \in P(Q)$

δ' = 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
= $\{Q_i \mid \underbrace{Q_i \in P(Q)}_{Q_i \in P(Q)=Q'} \text{ and } Q_i \cap F \neq \emptyset\}$

= (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

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

$E^*(A)$

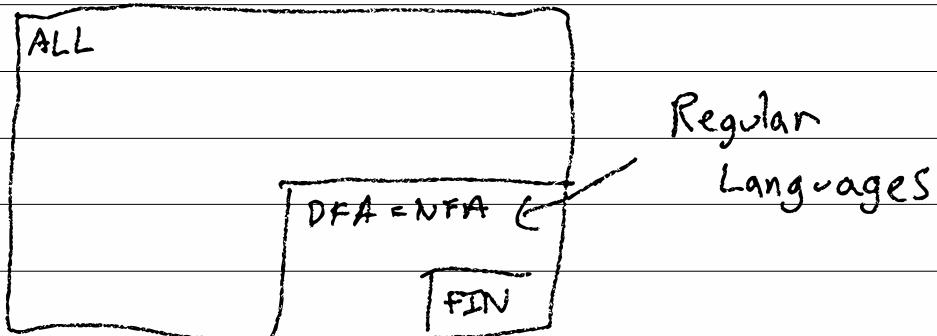
$E^*(A) = A \cup \bigcup_{g_i \in A} \delta(g_i, \varepsilon)$

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

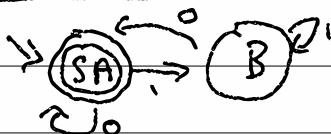
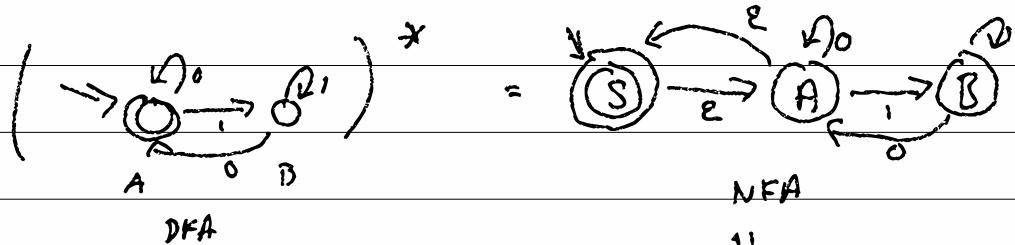
$$N = D$$

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

$$D = N$$



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



6-2 / Regular Expressions

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

interface RegEx { }

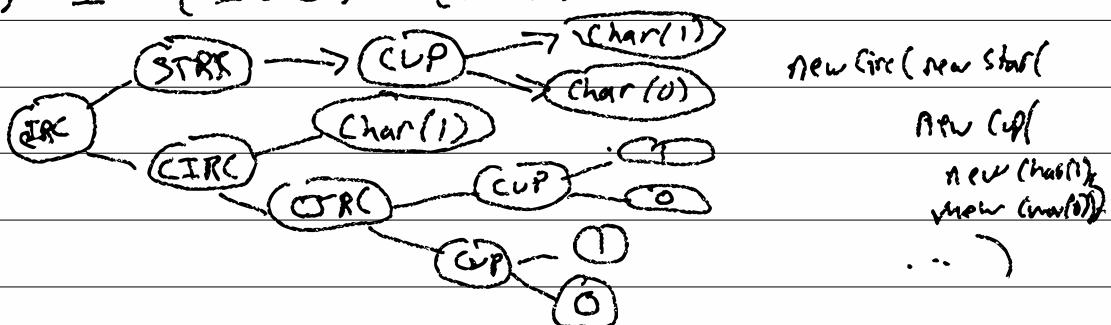
class RE_EMPTY() impl REGEX { .. . }

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)

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

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

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

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

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

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

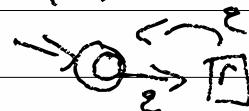
class Up {
 $L()$ \in
 return up(lhs, L(), rhs, L());
}

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)

$$\bullet = \epsilon$$

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

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 \Leftarrow

304.c

← ↗ ↘

304.c main.c

dir

304.c

awesome-quotes.txt

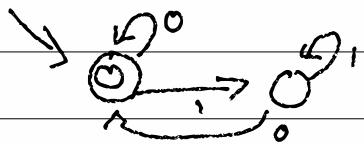
mycat.jpg

main.c

+

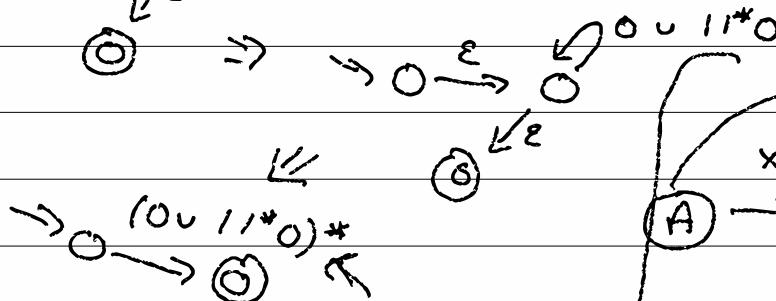
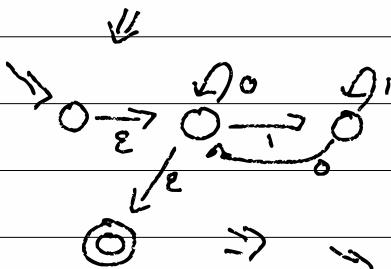
emacs

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



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

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

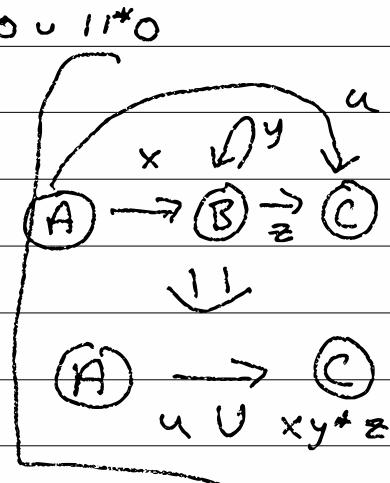


$$(0 \cup 11^* 0)^*$$

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

011101111100

11100011101100101010



G-9 / decompile : N-state NFA
→ RE

START : N-NFA → (N+2)-GNFA

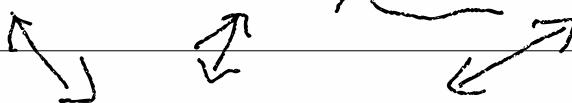
RIP : (N+1)-GNFA → N-GNFA

END : 2-GNFA → RE

decompile m = end ∘ ripⁿ ∘ start (n)

7-1) DFA/NFAs \rightarrow RE

DFA_s \leftrightarrow NFA_s \leftrightarrow RE



Regular
Languages

NFA \rightarrow RE

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

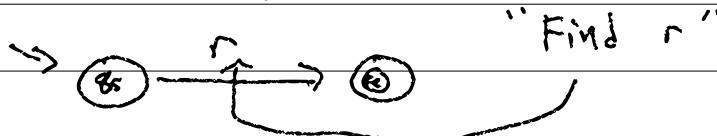
RIPⁿ: (n+1)-GNFA \Rightarrow n-GNFA

OUT: 2-GNFA \rightarrow RE

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

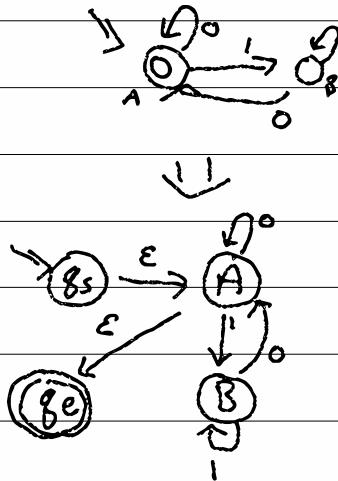
OUT: 2-GNFA \rightarrow RE

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



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

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



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

$$\Delta(g_s, g_o) = \epsilon$$



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

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

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

$$\Delta(g_i, g_j) = \cup \{\epsilon_{\text{for } g_e}\}$$

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

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

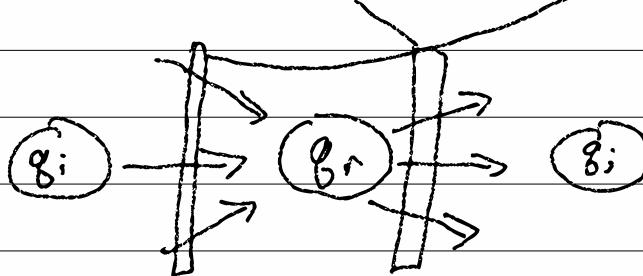
↓
 jay
 main

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

↓
 jay
 main
 ↓
 exit
 ↓
 exit

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

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

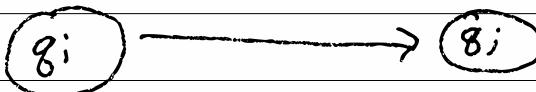


$$x \circ y^* \circ z \cup \alpha$$

$$\rightarrow x \circ z$$

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

$$=$$

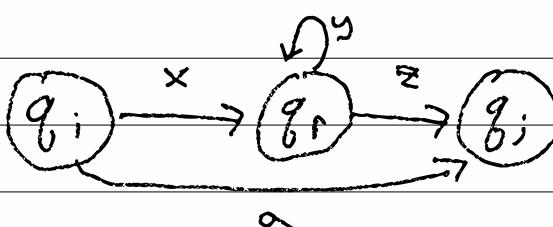


$$x = \Delta(q_i, q_r)$$

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

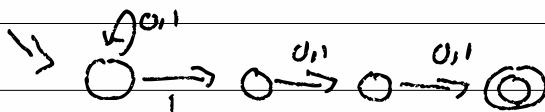
$$z = \Delta(q_r, q_j)$$

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

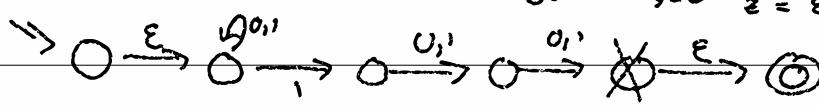


7-4/ int main () {
 int x = 8;
 int y = f(x, z);
 ...
 int y = x + 2 * z;

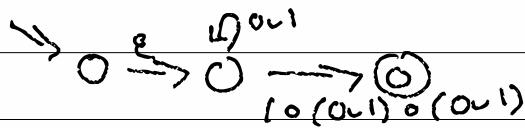
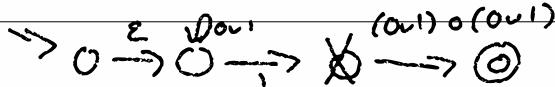
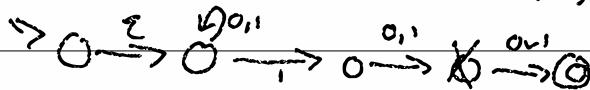
int f(int x, int z) {
 return
 z + 2 * x; } }



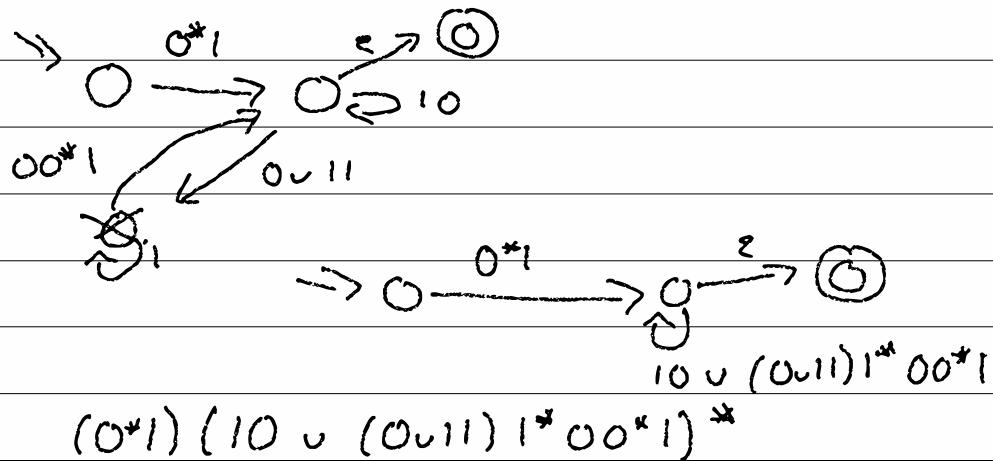
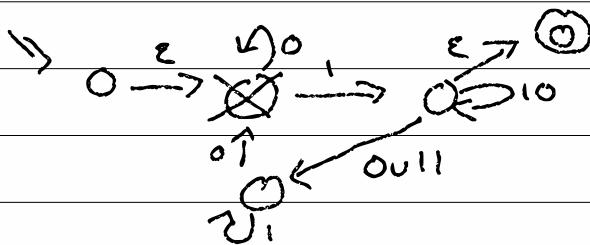
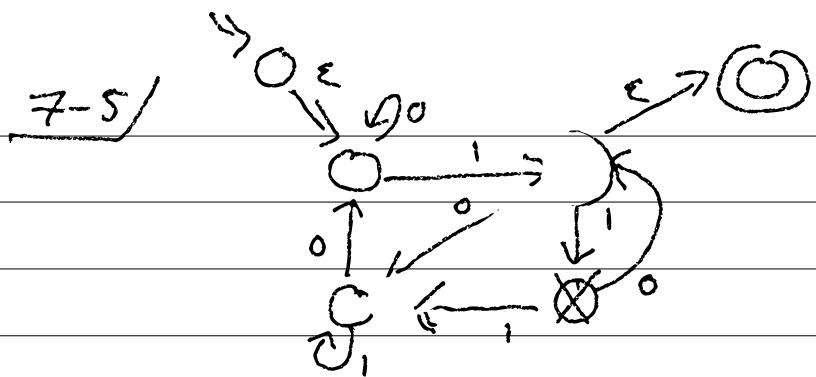
$$\text{out} = x \quad y = d \quad z = e \quad a = \emptyset$$



$$x \circ y \circ z \circ v \circ a = (x \circ y \circ z \circ v) \circ a \\ = (x \circ y \circ z \circ v) \circ a$$

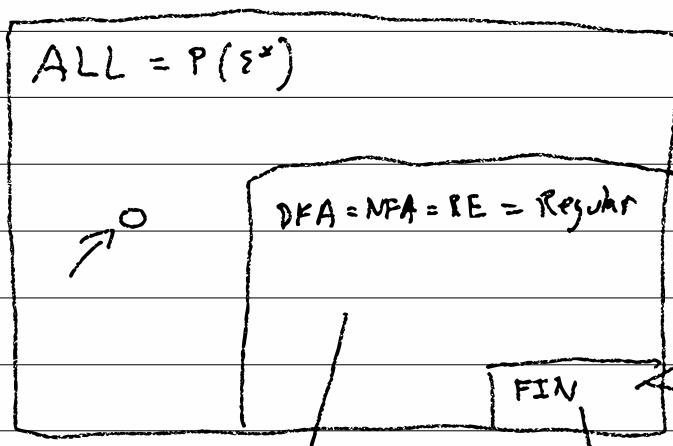
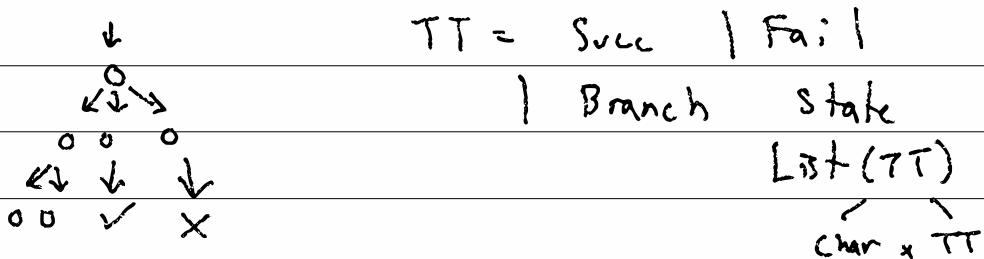


$$\rightarrow 0 \longrightarrow \textcircled{0} \quad \epsilon o (out) * o 1 o (out) o (out) = \epsilon * 1 \leq \infty$$



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

TT



- $A \in REG \iff \exists d \text{ DFA. } L(d) = A$
- $\neg A \in REG \iff \neg (\exists d) \iff \forall d. d \text{ DFA} \wedge$
- $\neg \exists x. P(x) \iff \forall x. \neg P(x) \quad L(d) \neq A$
- $\neg \forall x. P(x) \iff \exists x. P(x)$

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

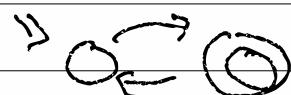
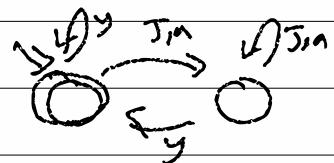
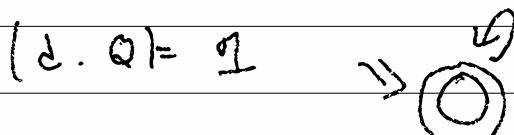
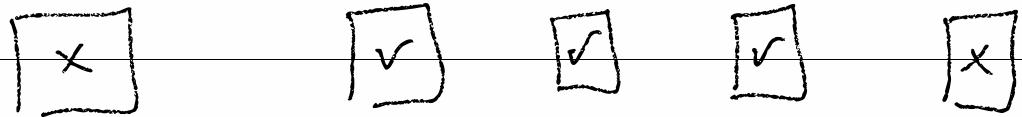
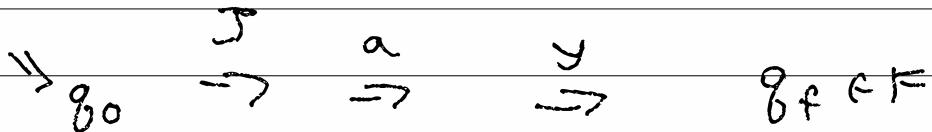
conclude $B \notin \text{REG}$.

$$\Sigma = \dots$$

8-3)

"Jay" & d

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



Daphne wins

pick a number of states ; 4

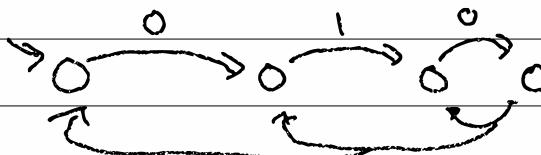
she picks a char

I say what state we goto

I win if I never say same state

she wins if I repeat

How many turns to win?

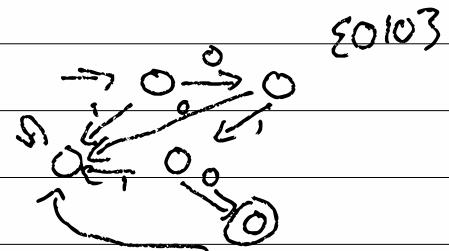
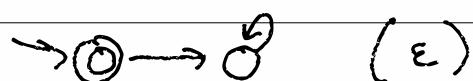


$4 \Rightarrow N \Rightarrow$

|@|

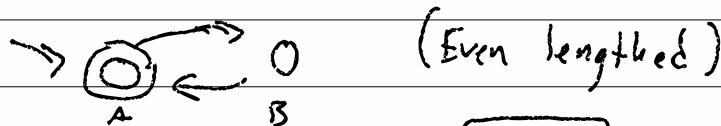
$\delta: Q \times \Sigma \rightarrow Q$ Total Fun
 $\xrightarrow{\text{from}} \xrightarrow{\text{to}}$

All DFAs have a loop



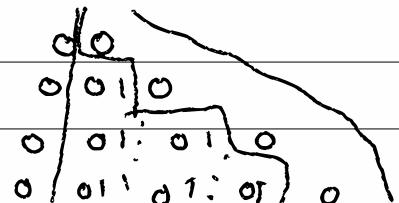
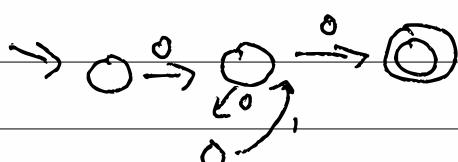
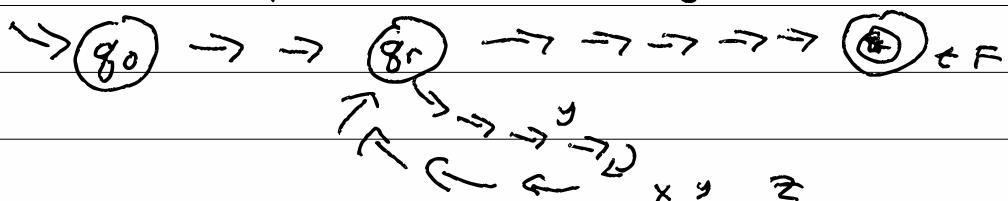
Some have "exciting" loops

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



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

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



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

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

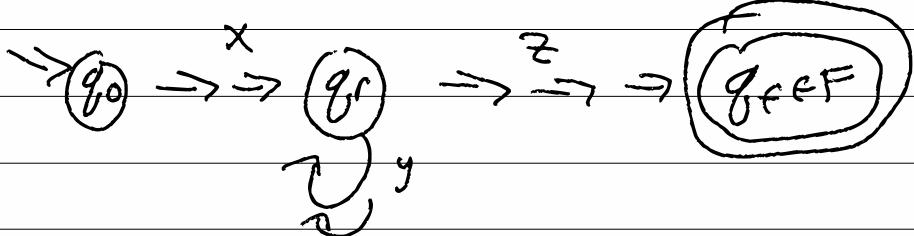
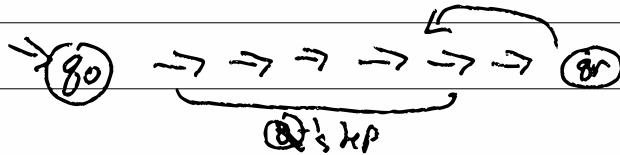
If it does have one ...

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

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

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

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

$|xy| \leq p$

$|y| > 0 \quad)$

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

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

)

$\forall d \in \text{DFA}, \exists r \in \text{RE}. \quad L(d) = L(r) \quad - \text{Cof}$

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

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

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

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

$\exists i \in \mathbb{N},$

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

8-7) Need: an infinite space problem

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

$$= 2^{32} \text{ uint2_t}$$

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

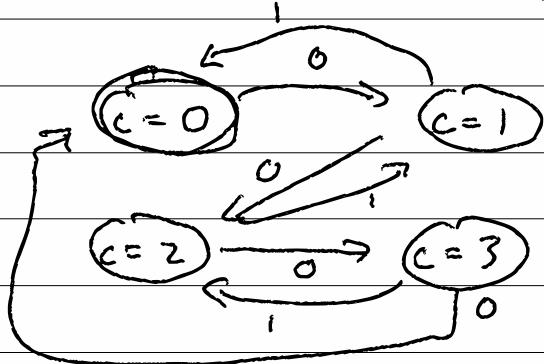
$0^n 1^n \in B$

$\text{uint} \text{ count} = 0;$

while ($in == 0$) count++

while ($in == 1$) count--
if ($in == \text{EOF}$) \leftarrow count--
return count ≥ 0 }

{ net false } }



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

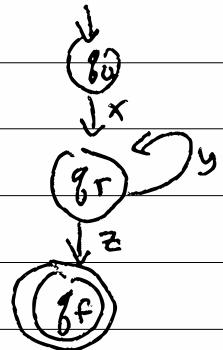
$$\exists p \in N, \quad |P| = |Q|$$

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

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

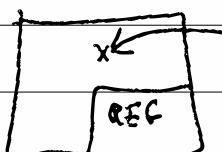
$$\forall i \in N, \quad |xy| < p$$

$$xyiz \in A \quad |y| > 0$$



$\neg RPP(A) =$

ALL



$$\forall p \in N,$$

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

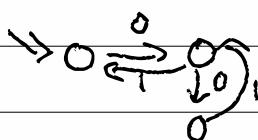
$$\forall (x, y, z \in \Sigma^*) \mid s = xyz \wedge |y| < p$$

$$\exists i \in N$$

$$xyiz \notin A$$

$$0^n 1^n \quad \text{ie} \quad x \in 0^n 1^n$$

$$\text{iff } \exists n \in N, \quad x = 0 \underbrace{\dots}_\text{long part} 0 1 \dots$$

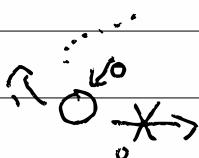


$$\text{int count} = 0$$

while (see 0) count++

while (see 1) count--

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 $s: s = x = z$

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

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

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

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

$|y| > 0$)

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

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

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

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

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

$a+b+c = d$

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

fun $n:$

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

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

else: let $m = n-1$

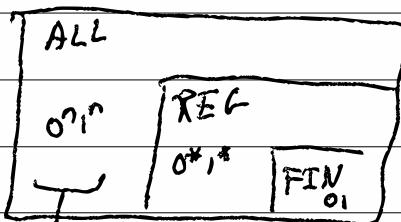
$p \circ : \text{th. } p_n \rightarrow p_{(n+1)}$

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?



All computers
are DFAs.

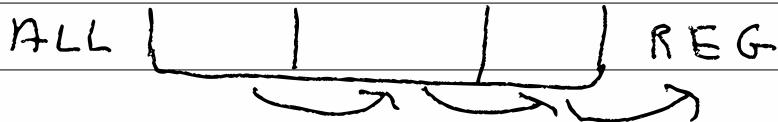
infinitely big

if $0^n 1^n \in \text{REG}$

~~then~~^{and} $B \in \text{REG}$

then $0^n 1^n \circ B \in \text{REG}$

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



$0^n 1^n \in \text{REG}$

w where $\text{count}(0,w) = \text{count}(1,w)$

ww where $w \in \Sigma^*$

010101 $\in \Sigma^*$

wwR where $w \in \Sigma^*$

$$\underline{q-y} / \quad 0^x 1 0^y 1 0^{x+y}$$

$$01\ 0100 \Rightarrow "1+1=2" \quad \checkmark$$

$$001\ 0001\ 00000 \Rightarrow "2+3=5" \quad \checkmark$$

$$001\ 0010 \Rightarrow "2+2=1" \quad \times$$

A.p.

$$\exists s . \quad 0^p 1 0^p 1 0^{2p}$$

$$\nexists x y z . \quad x = 0^a \quad y = \underline{0^b} \quad z = 0^c 1 0^p 1 0^{2p}$$

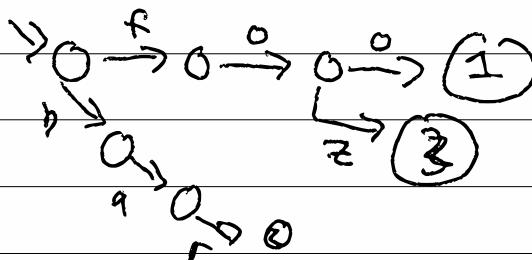
$$\exists ; . \quad \begin{matrix} \text{change this} \\ \text{can't change} \end{matrix}$$

$$\underbrace{z+z=4}$$

$$\Rightarrow \Rightarrow$$

$$\underbrace{g+z=10}$$

12-1 / trie $\text{foo} \leftrightarrow 1$ digital
 $\text{bar} \leftrightarrow 2$ search
 $\text{baz} \leftrightarrow 3$ tree



CFGs

start variable $0^n 1^n$ & REG

$S \rightarrow \epsilon$ ← rule, productions, transitions
 D $S \rightarrow OSI$

variables
non-terminals

Symbols

rule = $\overbrace{V}^{\text{lhs}} \rightarrow \overbrace{V \cup \epsilon}^{\text{rhs}}$
 $\text{rhs} = (\text{V} \cup \epsilon)^*$
 $\xrightarrow{\text{terminal}}$

$\overset{B}{S} \rightarrow \overset{B}{OSI} \rightarrow \overset{B}{0OSII} \rightarrow \overset{B}{000SIII}$
 "derivation" $\overset{B}{000III}$

$w \in L$ iff $\exists d. S \xrightarrow{*} w$

$$\frac{L(2)}{M} \left(\begin{array}{l} S \Rightarrow 01S \\ S \Rightarrow SS \end{array} \right) = \emptyset$$

Context-free grammar

$$0101 \in \xrightarrow{\quad} \text{alphabet} \xrightarrow{\quad} (V, \Sigma, R, S)$$

$$B \Rightarrow \epsilon \qquad \qquad \downarrow \text{finite set}$$

$$B \Rightarrow B \cup N \cup B \qquad \qquad \downarrow \in V$$

$$N \Rightarrow \epsilon \qquad \qquad P(V \times (V \cup \Sigma)^*)$$

$$N \Rightarrow 0N \qquad \qquad (V \Rightarrow P(V \cup \Sigma)^*)$$

$$\Sigma(S, 01S),$$

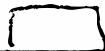
$$R = \{S, SS\} \quad V = \{\Sigma\} \quad \Sigma = \{0, 1\} \quad S = S$$

REG

DFA

REX ($\cup, \cdot, ^*$)

CFL



CFG

12-3 / A B

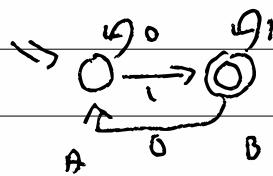
$\rightarrow A \cup B : S \Rightarrow A, S$

$S \Rightarrow B, S$

$A \circ B : S \Rightarrow A, S \quad B, S$

$A \cap B : X$

DFA \Rightarrow CFG



01101

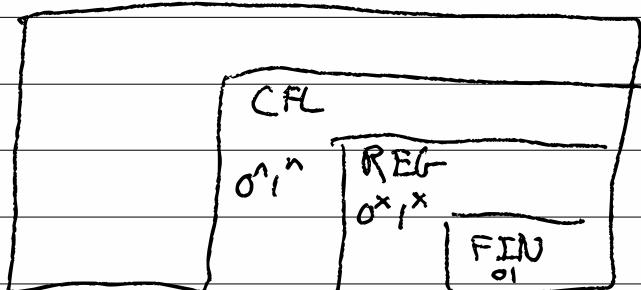
$S = g_0$

$g_i \rightarrow c g_j \text{ iff } \delta(g_i, c) = g_j$

$g_f \rightarrow \epsilon \text{ iff } g_f \in F$

x	ϵ	
$A \rightarrow 0A$	$ $	1B
$B \rightarrow 1B$	$ $	0A ε
\vdash	w	v

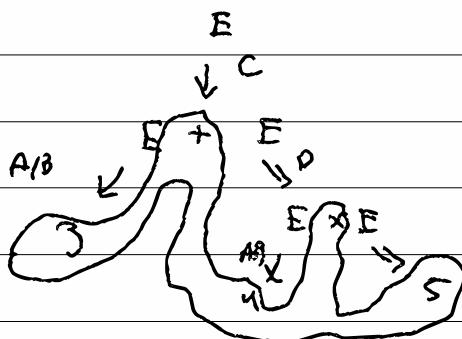
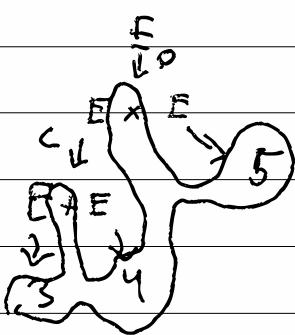
$A \xrightarrow{x} 0A \xrightarrow{\epsilon} 01B \xrightarrow{\vdash} 011B$
 $\xrightarrow{w} 0110A \xrightarrow{\epsilon} 01101B \xrightarrow{v} 0111$



P D C D

$$12 - 4 / E \Rightarrow O \quad | \quad I \quad | \quad E + E \quad | \quad E \times E$$

$$3 + 4 \times 5$$



fowim $E \Rightarrow n$

fowin $O = O$

$I = I$

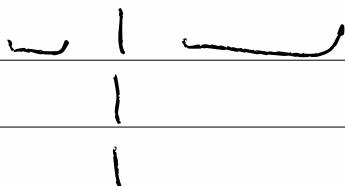
Plus(L, R) \Leftarrow fowim L + fowim R

Mult(L, R) \Leftarrow fowim L \times fowim R

ambiguous = There are multiple
trees (derivations)

if cond

for the same string



(2-5) Nice! amb \rightarrow unamb

not possible

V, Σ, R, S

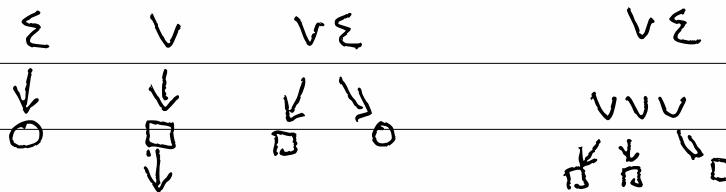
$$V \Rightarrow (V \cup \Sigma)^*$$

ϵ

$\epsilon \Sigma V$

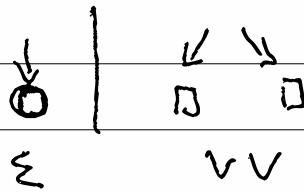
V

VVVVV



want: "pure" binary trees

Noam



Chomsky-Normal Form

$$\begin{array}{c|c} V \rightarrow \Sigma & A \rightarrow C \\ V \rightarrow VV & A \rightarrow BC \\ S \rightarrow \epsilon & \#S \end{array}$$

GFG \rightarrow CNF

(2-5) $S \rightarrow \epsilon \mid OSI$

— Add a new start

$R \rightarrow S$

$S \rightarrow \epsilon \mid OSI$

— remove ϵ -rules

$R \rightarrow S \mid \epsilon$

$S \rightarrow OSI \mid OI$

remove unit rules

$R \rightarrow OSI \mid OI \mid \epsilon$

$S \rightarrow OSI \mid OI$

add extra vars for ≥ 2

$R \rightarrow TI \mid OI \mid \epsilon$

$S \rightarrow TI \mid OI$

$T \rightarrow OS$

add terminals "names"

$R \rightarrow TB \mid AB \mid \epsilon$

$S \rightarrow TB \mid AB$

$T \rightarrow AS$

$A \rightarrow O$

$B \rightarrow I$

12-6 / 00111 + 0ⁿ1ⁿ

S $\Rightarrow \epsilon$

S $\Rightarrow 0S1$

A

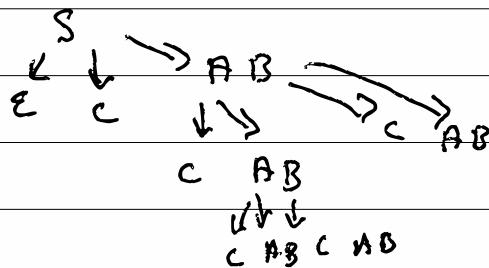
B

A $\Rightarrow c$ + 1

B \Rightarrow + 2

A $\Rightarrow BC$ + 2

A \Rightarrow done



S $\Rightarrow XYZ$

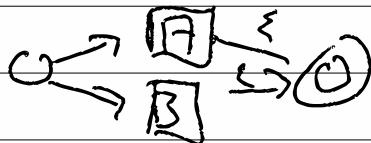
X $\Rightarrow ZZS$ | 0

Y $\Rightarrow XYX$ | 1

Z $\Rightarrow YSX$

$$A = \{x, y, z\}$$

$$B = \{x, y, z\}$$



$$Q = \{S, E, (0, x), (0, y), (0, z), (1, x), (1, y), (1, z)\}$$

$$= \{S, E\} \cup 0 \times A \cup 1 \times B$$

unionstate = start | end | fromA g_A
| fromB g_B

$$\delta(\text{start}, \epsilon) = \{ \text{fromA } g_0, \text{ fromB } g_0 \}$$

$$\delta(\text{fromA } g_i, c) = \text{fromA } (\delta_B(g_i, c))$$

12-7 / new DFA ("blast",