

L-1 effective math  $\sum_{i=0}^{20} z_i \text{ vs } \sum_{i=0}^{\infty} z_i$

$$9x^3y^2z + 8xyz - 99xy^2z^4 = 0$$

which statements are true?

"All birds have wings"

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

Defining the set of true statements

Making a decision procedure

Generating a list

---

A statement is a string of characters from an alphabet  
some finite set  $\Sigma$

A finite set is one where you can write down all elements

all the elements:  $S = \{ \text{Pikachu, Charmander, Squirtle, Bulbasaur} \} = \{ C, P, B, S \}$

A string of  $\Sigma$  is a sequence of  $\Sigma$   
 $P P P P$        $C S C S C S S \underbrace{S}_{} = \epsilon$

1-3 A language is a set of strings  
 $\{\epsilon, P, PP, PPP, PPPP\}^*$  - finite       $\{P, P^2, \dots, P^{256}, \dots\}$

$x \in S$  -  $x$  is inside  $S$        $P \in \{\epsilon, P, PP\}$

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

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

$x \in \bar{Y}$  iff  $x \notin Y$  (but  $x \in U$  - universe)

→ complement or negation of  $Y$

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

$PP \circ BC = PPBC$

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

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

$PPCE$

is a group

lexicographic ordering of  $\Sigma$

$lo(\Sigma_0, 13) = \underbrace{\epsilon, 0, 1, 00, 10,}_{600, 001, 010, 011, 100, 101, 110, 111}$  ...

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

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

1-3)  $\text{lexi} : \text{num} \rightarrow \text{string of } \Sigma \quad |\Sigma| = 2$

$\text{lexi } 0 = \epsilon \quad \text{lexi } 1 = 0 \quad \text{lexi } 2 = 1$

$\text{lexi } 3 = 00$

$\text{lexi } n = \text{size of } \Sigma$

if  $n < \text{size}^0$  then ret  $\epsilon$  often

$(n - \text{size}^0) < \text{size}^1$  then convert  $(n - \text{size}^0)$  into  $\epsilon$

$(n - \text{size}^0) - \text{size}^1 < \text{size}^2$  convert of len 2

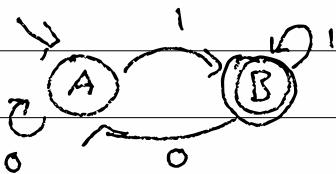
The set of strings in the lexicographic ordering

of  $\Sigma \approx \Sigma^*$

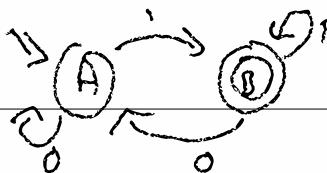
$\epsilon \in \Sigma^* \quad \text{PPCBSPPPP} \in \Sigma^*$

$010111 \in \Sigma^*$

Deterministic Finite Automata (DFA)



## 2-1) DFA



$\circlearrowleft$  means loss

$$0110 = \text{No}$$

1 = Yes

$\circlearrowright$  means No

$$0111 = \text{Yes}$$

11 = Yes

$$0010 = \text{No}$$

00 = No

transition function (edges)

$$1101 = \text{Yes}$$

$Q \times \Sigma \rightarrow Q$

$$\varepsilon = \text{No}$$

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

$A \mid B$	always finite are the states	$\mid$	$\mid$	$\mid$
$\begin{array}{ c c } \hline 0 & A \\ \hline 1 & B \\ \hline \end{array}$	$\{A, B\}$	$\{0, 1\}$	startstate $= A$	accepting states $\{B\}$

"n % 2 == 1" if "odd? n"

No string DFA :

only empty string :

only the string 'J' :

'Ja' and 'Jb'

$: Q \times \Sigma \rightarrow Q$

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

accepts?  $d \mid s : \text{DFA} \times \Sigma^* \rightarrow \text{Bool}$

accepts?  $d \mid \varepsilon = \text{is } q_0 \text{ in } F:$

$d.F, \text{member}(d, q_0)$

accepts  $d \mid (c :: s)$

$: \text{DFA} : Q : \Sigma^*$

accepts  $d \mid s = \text{helper } d \mid d \cdot q_0 \mid s$

helper  $d \mid q_i \mid \varepsilon = q_i \in d.F$

helper  $d \mid q_i \mid (c :: s) = \text{helper } d \mid q_i \mid s$

$q_j = d.\delta(q_i, c)$

DFA::Accepts ( 304 string s ) {

$Q \mid q_i = \text{this} \cdot q_0;$

while ( s != empty ) {

$q_i = \text{this}, \delta(\text{delta}(q_i, s.\text{first}))$ ;

$s = s.\text{rest}$  }

return  $\text{this}, F, \text{member}(q_i)$  }

↙ trace

2-3) 0110  $\Rightarrow$  Even, Odd, Odd, Even

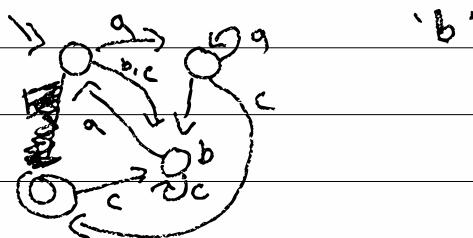
Transducers are DFAs where state writers  
Moore machines

$L(d)$  = the language of DFA  $d$   
 $= \{ s \mid \text{accepts } d \text{ } s = \text{true} \}$   
may be infinite

Given a DFA, return a string that would be accepted

example : DFA  $\Rightarrow \Sigma^*$  or false

s.t. If example  $d$  returns  $s$  then  
accepts?  $d \text{ } s = \text{true}$



2-y Suppose that  $d$  is a DFA, construct  $d'$  where

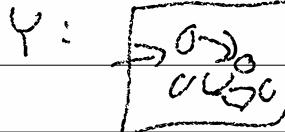
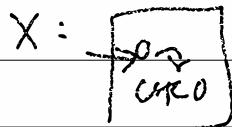
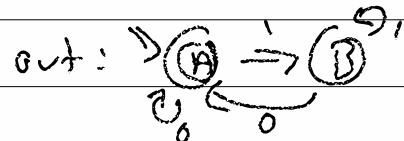
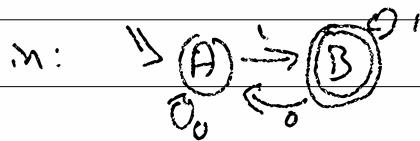
$$L(d') = \overline{L(d)} \quad (\text{ie } d' \text{ says } s \text{ yes})$$

negate: DFA  $\rightarrow$  DFA

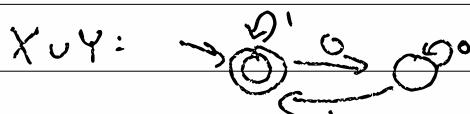
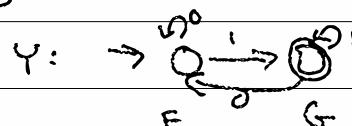
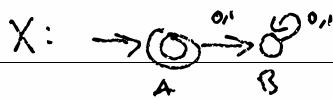
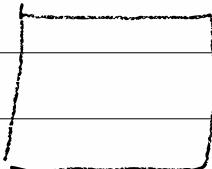
when  $d$  says no

negate (tddcs) = Evens

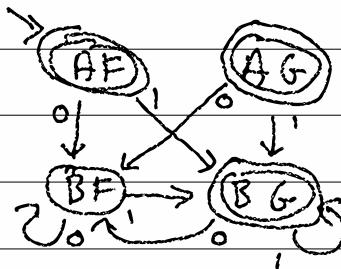
& vice versa)



$X \cup Y$ :



Z:



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

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

$$z \cdot \Sigma = x \cdot \Sigma = y \cdot \Sigma$$

$$z \cdot g_0 = (x \cdot g_0, y \cdot g_0)$$

$$z \cdot F = \{ (g_x, g_y) \mid g_x \in x \cdot F$$

(or)  $g_y \in y \cdot F \}$

$$z \cdot \delta((g_x, g_y), c)$$

$$= (x \cdot \delta(g_x, c),$$

$$y \cdot \delta(g_y, c))$$

and to make  
intersect

$$X \subseteq Y \text{ (subset) iff }$$

$$\forall g \in X, g \in Y.$$

$$X = Y, \text{ iff }$$

$$X \subseteq Y$$

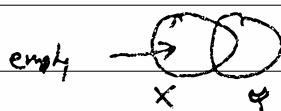
$$\text{and } Y \subseteq X$$

subset? : DFA  $\times$  DFA  $\rightarrow$  bool

subset? ( $\Downarrow_{\text{DFA}}$ )  $X = \text{Yes}$

(epsilon) (Even) = Yes

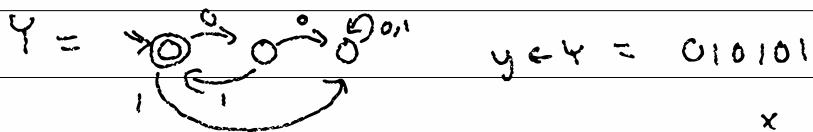
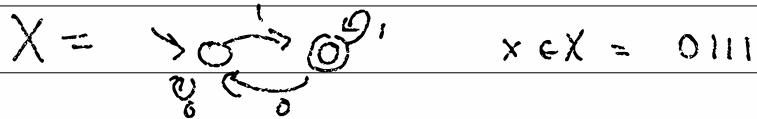
(epsilon) (odd) = No



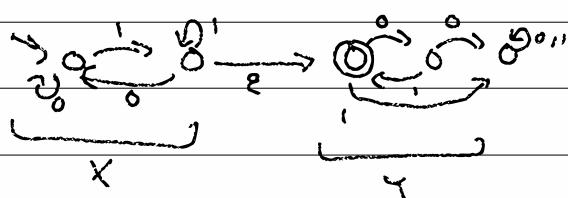
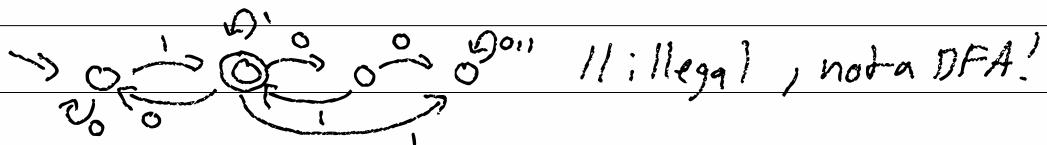
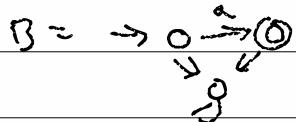
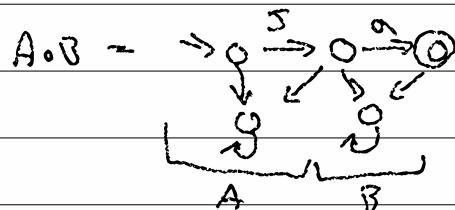
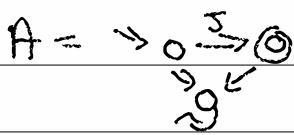
$X - Y$  must be empty

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

3-1  $z \in X^0 Y$  iff  $z = xy$  where  
 $x \in X$  and  $y \in Y$



$$z \in X^0 Y = \overbrace{0111}^x \overbrace{010101}^y$$

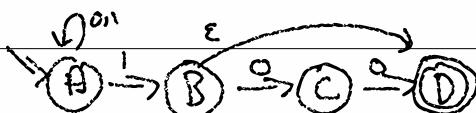


$F \xrightarrow{0 \rightarrow 0} G$   
 (skip from F to G  
 at the right  
 time)

3-2) NFA - non-deterministic finite automata

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

$$NFA = (Q, \Sigma, q_0, \delta: Q \times \Sigma \underset{\text{sigma events}}{\rightarrow} Q, F \subseteq Q)$$
$$\rightarrow P(Q)$$



S	A	B	C	// D
0	ΣA3	(C3	ΣD3	Σ3
1	ΣAB3	Σ3	Σ3	Σ3
ε	Σ3	ΣD3	Σ3	Σ3

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

$x \notin L(n)$  iff

forall oracle  $n \neq$  ~~for~~ (or "if")

trace = a sequence of  $Q \times \Sigma \cup \epsilon$

$$(A,0) (B,1) (C,0) (D,0) \quad 0100 \in L(n)$$

$$(A,1) (A,0) (B,1) (D,\epsilon) \quad 101\epsilon = 101 \in L(n)$$

oracle interpretation : NFA  $\times$  trace  $\rightarrow$  boolean

oracle  $N + =$  helper  $N \cdot q_0 +$

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

$((q_i, c) :: +') =$

is  $q_i \in N \cdot \delta(q_i, c)$ , then helper  $N q_i +'$   
o.w. "invalid trace"

3-3 / trace-tree = accept | reject  
 | branch state (++, ...)

all : NFA  $\times \Sigma^* \rightarrow \uparrow$   
 ↑  
 set

all  $N$   $s$  = helper  $N$   $N, g_0$   $s$

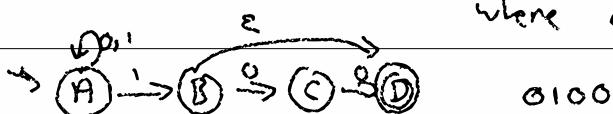
helper  $N$   $g_i$   $s$  =

branch  $g_i$  (case  $s$  where  
 $\{ \} \rightarrow$  if  $g_i \in N, F$  then  
 $\{ \text{accept} \}$   
 o.w.  
 $\{ \text{reject} \}$ )

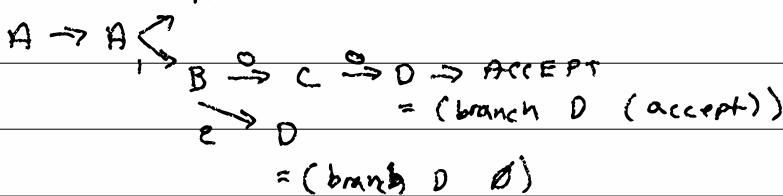
$c :: s' \rightarrow$

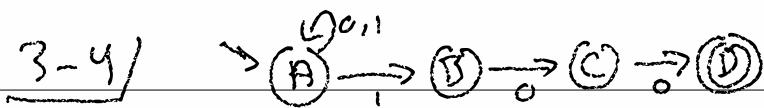
$\{ \text{++} \mid ++ = \text{helper } N g_j s'$   
 where  $g_j \in N, \delta(g_j, c) \}$

$\cup \{ \text{++} \mid ++ = \text{helper } N g_j s$   
 where  $g_j \in N, \delta(g_j, \epsilon) \text{ and } g_j \neq g_i \}$



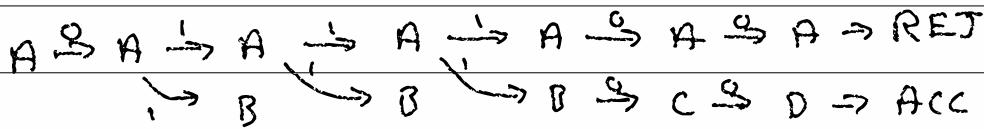
$0 \quad 1 \xrightarrow{A} \xrightarrow{B} \xrightarrow{C} \xrightarrow{D} \text{Reject}$





"ends in 100"

011100



backtrack : NFA  $\times$  String  $\Sigma^*$   $\rightarrow$  Bool

backtrack  $N$   $s = \text{helper } N \ N.g_o \ s$

helper N g; s =

OR case 5 with  $\square \rightarrow g_i \in N, F$

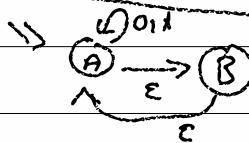
$c::s' \rightarrow OR_{g_j \in N.S(g_i, c)} helper(N.g_j.s')$

OR

$$g_j \in N_\epsilon(g_i, \epsilon)$$

helper N g; s

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



maybe DS = tree

unfold :  $A \rightarrow DS(B)$

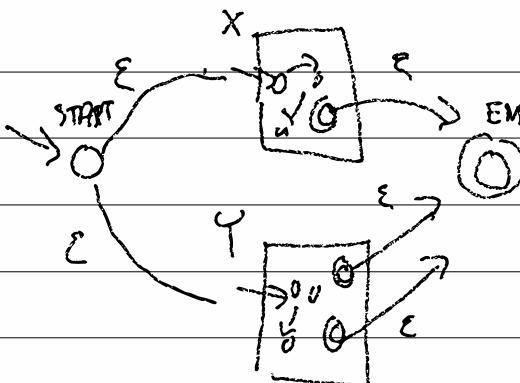
$$\text{fold} : \text{DS}(B) \rightarrow C$$

Haskell

there always exist

combined : A  $\Rightarrow$  C

4-1/



$$X = (Q_X, \Sigma, g_{0x}, \delta_X, F_X)$$

$$Y = (Q_Y, \Sigma, g_{0y}, \delta_Y, F_Y)$$

$$Z = (Q_Z, \Sigma, g_{0z}, \delta_Z, F_Z)$$

$$F_Z = \{\text{END}\}$$

$$g_{0z} = \{\text{START}\}$$

$$Q_Z = \{\text{START, END}\}$$

$$\delta_Z(g_i, c) =$$

$$\cup Q_X \times \{\emptyset\}$$

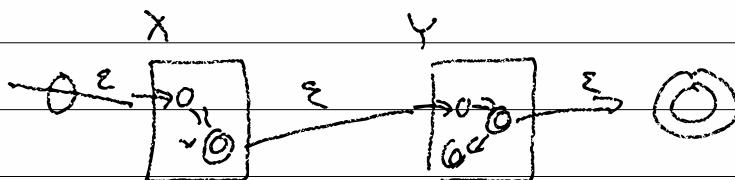
$$(\text{START}, \epsilon) = \{(g_{0x}, 0), (g_{0y}, 1)\} \quad \cup Q_Y \times \{\emptyset\}$$

$$(\text{START}, -) = \{\emptyset\}$$

$$(\text{END}, -) = \{\emptyset\}$$

$$((g_{0x}, 0), c) = \delta_X(g_{0x}, c) \times \{\emptyset\}$$

$$g_{0y}, \text{ similar} \quad \cup \text{ if } g_{0x} \in F_X \text{ and } c = \epsilon, \{\text{END}\} \text{ o.w. } \{\emptyset\}$$



4-2) Kleene-star  $X^*$

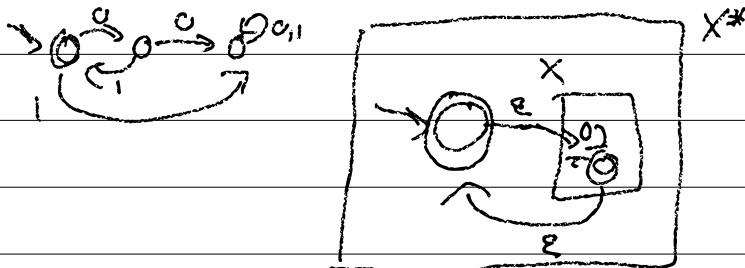
$\bar{z} \in X^*$  iff  $\bar{z} = z$  OR  $\bar{z} = xy$

where  $x \in X$  and  
 $y \in X^*$

iff  $z = x_0 \dots x_n$

where  $x_i \in X$

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

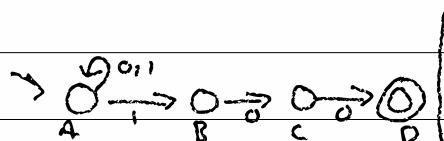


DFA's =  $\cup, \cap, -$

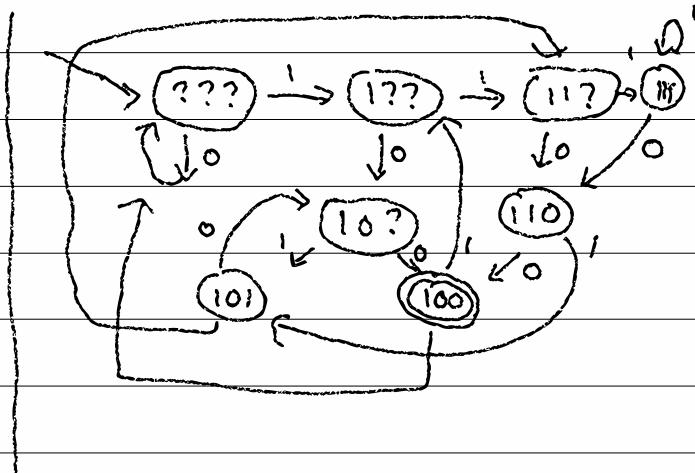
## DEA $\rightarrow$ NFA

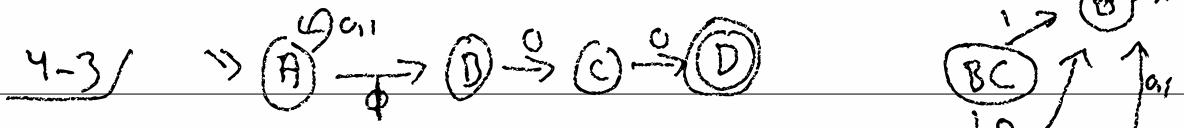
NFAs :  $\cup$ ,  $\circ$ , \*

wayf: NFA  $\Rightarrow$  DFA

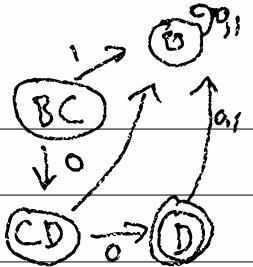


0100





$01100$   
 $A \xrightarrow{\delta} A \xrightarrow{\circ} A \xrightarrow{\circ} A \xrightarrow{\delta} A$   
 $B \xrightarrow{\circ} B \quad C \quad D \leftarrow \checkmark$



???

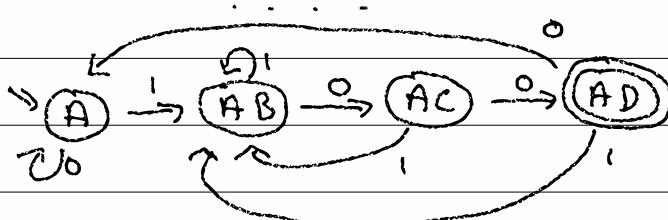
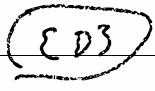
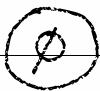
???

??

1111

1102

100



01100 ✓

011001 X

1111 X

111100 ✓

NFA  $\Rightarrow$  DFA    in :  $(Q_N, \Sigma, \delta_N, F_N)$

out :  $(Q_D, \Sigma, \delta_D, F_D)$

$$Q_D = P(Q_N) \quad \delta_D = \{ \delta_{nq} \mid q \in Q_D \}$$

$$\delta_D(q_D, c) = \bigcup_{q_n \in q_D} \delta_N(q_n, c)$$

~~q\_n \in q\_D~~ ~~c \in~~

$$E : Q_D \rightarrow Q_D = E(\ast) = \bigcup_{q_D} \bigcup_{q_n \in q_D} \bigcup_{c \in \Sigma} \delta_N(q_n, c)$$

Q\_M

$\forall x \in \Sigma^*$ , exec backtrace  $N \ x$

     = accepts ( $NFA \rightarrow DFA \ N$ )  $x$

random string

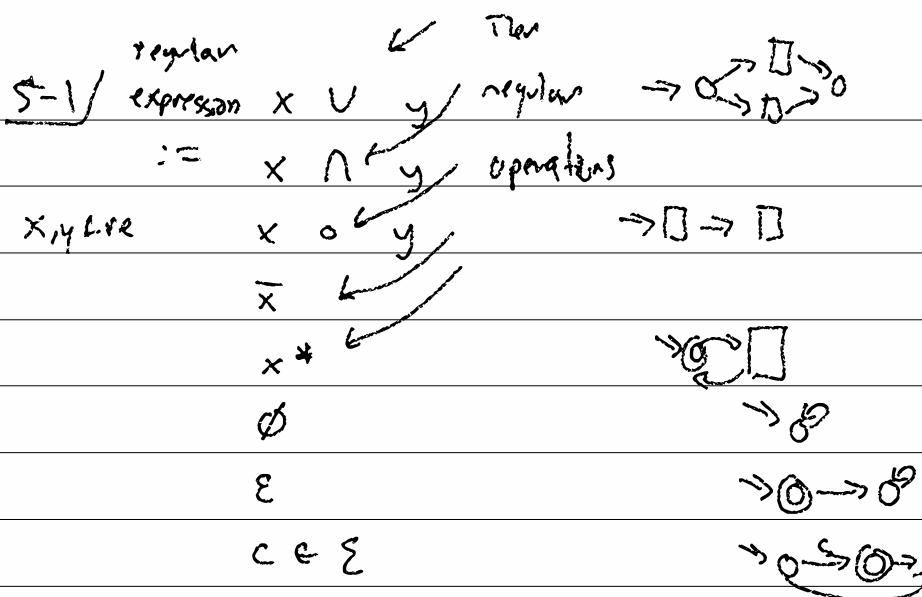
= pick a random number

(lex; n)

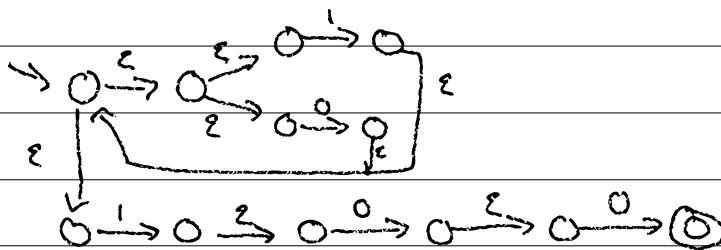
write the NFA manually as a VKA...

use the dfa equality checker to see  
that

dfa-equal? manual ( $N \Rightarrow D \ N$ ) = #true



$$(1, 0, 0, 1)^* (0, 1, 0)^* = (1, 0)^* 100$$



=

$$\xrightarrow{0} \square \xrightarrow{1} \square \xrightarrow{0} \square \xrightarrow{0} \square \xrightarrow{0} \square \xrightarrow{\varepsilon} \square \xrightarrow{0} \square \xrightarrow{0} \square \xrightarrow{0} \square \xrightarrow{\varepsilon} \text{final state}$$

$$* \cdot c \Rightarrow \varepsilon^* \circ 1 \circ 0 \circ 1 \circ c'$$

S-2/ interface RegEx;

class Empty implements RegEx; ()

Epstein              RegEx              ()

Char              RE              (char c)

Union              RE              (RE x, RE y)

Star              RE              (RE x)

Concat              RE              (RE x, RE y)

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

new Concat ( new Char('1'),

new Concat (new Char('0'),

=

new Char('0'))))]

(1v0)\*100

interface RegEx { NFA compile(); } }

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

S-3 / generate : RE  $\rightarrow \Sigma^*$  s.t.

accepts? (nfa $\Rightarrow$ dfa(compile r)) (generate r) = true

generate  $\emptyset$  = error

generate  $x \circ y$  = gen x  $\circ$  gen y

gen  $\epsilon$  =  $\epsilon$

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

gen c = c

gen ~~x~~  $\cup$  y = case (flip)

heads  $\rightarrow$  gen x

tails  $\rightarrow$  gen y

printall : RE  $\Rightarrow$  void

printall r = helper r print

helper  $\emptyset$  pr =  $\text{eg}(\text{void})$

h  $\epsilon$  pr = pr  $\epsilon$

h c pr = pr c

h  $x \circ y$  pr = h x newprint

(newprint s = h y npz

npz + = pr sat)

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

h  $x^*$  pr = h ( $\epsilon \cup x \circ x^*$ ) pr

h (xuy) pr = h x pr ; h y pr

numbers

$$\Sigma - y / \quad x \cdot 0 = 0 \quad x + y = y + x \quad x \cdot 1 = x$$

regex

$$x \circ \emptyset = \emptyset = \emptyset \circ x \quad \emptyset = \{\epsilon\}$$

$$x \circ \epsilon = x = \epsilon \circ x \quad \epsilon = \{\epsilon = "\"\}$$

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

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

$$\emptyset^* = \epsilon \quad x^* = \epsilon \cup x \circ x^*$$

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

$$(x^*)^* = x^* \quad \epsilon \cup \epsilon^* = \epsilon \cup \emptyset = \epsilon$$

$$x \cup z = z \text{ if } x \subseteq z$$


---

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



$$\xrightarrow{0} \xrightarrow{0} \xrightarrow{0} \xrightarrow{0} \Rightarrow (100)^* 100$$

IN

G-1/ NFA( $k$ )  $\rightarrow$  GNFA ( $2+k$ )

RIP  $\left[ \begin{array}{l} \rightarrow \text{GNFA } (2+k-1) \\ \rightarrow \text{GNFA } (2+k-2) \\ \dots \end{array} \right]$   $\xrightarrow{\text{k times}}$

 $\rightarrow$  GNFA (2)

OR

 $\rightarrow$  REGNFA:  $0 \xrightarrow{\delta} 0$ 

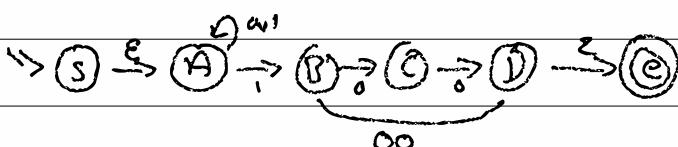
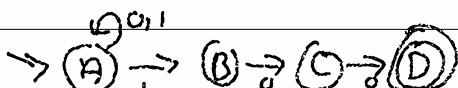
GNFA (generalized NFA)

 $0 \xrightarrow{\delta} 0$  $= (Q, \Sigma, g_a, \Delta, g_f)$ GNFA:  $0 \xrightarrow{\delta} 0$  $\uparrow$  one state, not a set $\Delta: (Q \times Q) \xrightarrow{\uparrow \downarrow} \text{Reg}$  $\delta: Q \times \Sigma \rightarrow P(Q)$  $(Q-g_f) \quad (Q-g_a)$ You can't  
leave  $g_f$ 

You can't

go back to  $g_a$ 

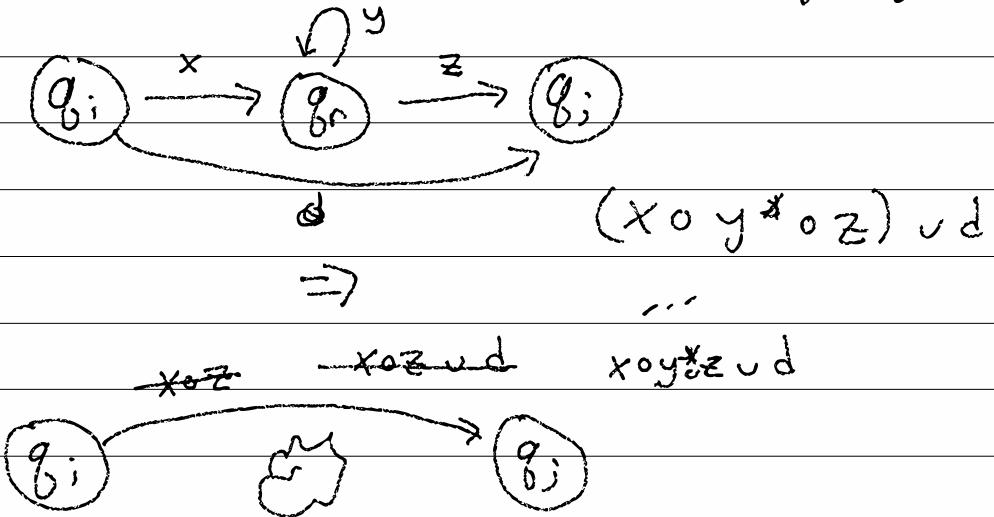
$\Delta(g_i, g_j) = r$   ~~$\rightarrow$~~   $x \in r$  then  $g_i \xrightarrow{x} g_j$  in the NFA



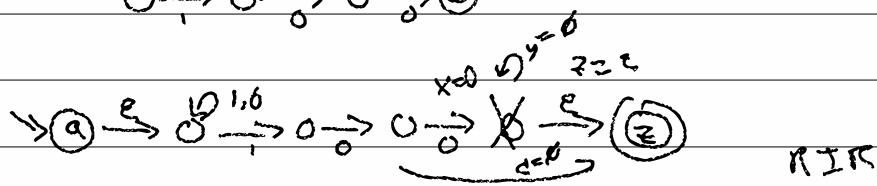
6-2/  $R_{10} : \text{GNFA } (n+1) \rightarrow \text{GNFA } (n)$

$\{q_0, q_f, q_r, q; \dots\} \rightarrow \{q_0, q_f, q; \dots\}$

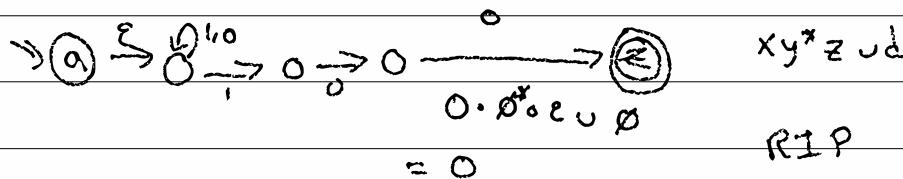
$q_r$  is gone



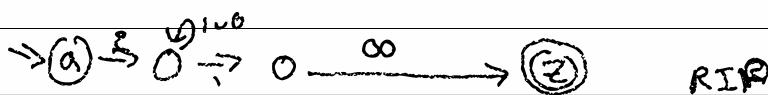
IN



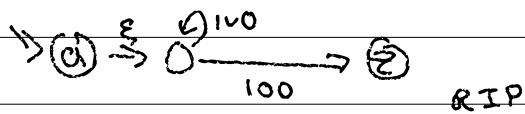
RIR



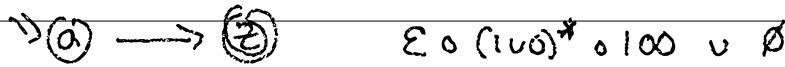
RIP



RIR



RIP

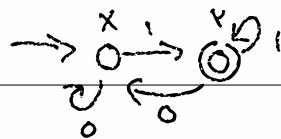


$$\Sigma 0 (1 \cup 0)^* \cdot 100 \cup \emptyset$$

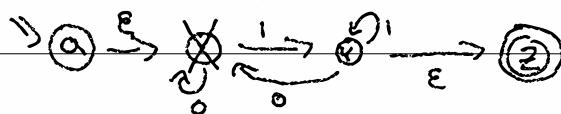
OUT

$$= (1 \cup 0)^* \cdot 100 \leftarrow$$

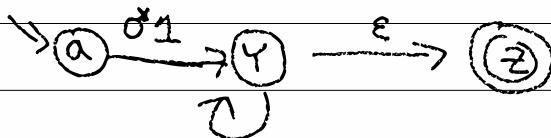
6-4)



↓ IN

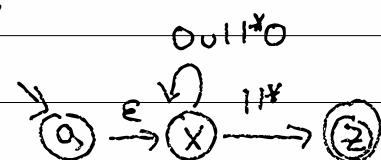


↓ RIP



$1 \cup 00^*1$

$$\text{↓ RIP} \\ 0^*1 \circ (1 \cup 00^*1)^* = //$$



↓ RIP

$(0011*0)^* 11^*$

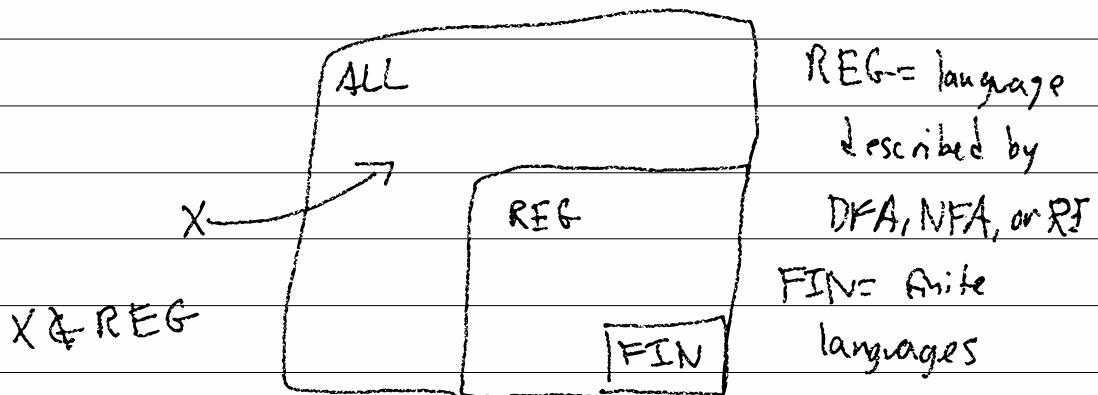
$(01)^*1$

DFA  $\leftrightarrow$  REX

Q.d. let  $n = \text{dfa} \rightarrow \text{rex}$  d

accepts? d (generate n) = true

G-5)



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

$$ALL = \Sigma^*, \quad \Sigma^* = \epsilon, 0, 1, 00, 01, 10, 11, \dots$$

$\{0, 1\}^*$ ,  $\{0, 00, \epsilon, 000, 0000, \dots\}$

{all of Jay's lectures}

{JPEGs of rats}, {JPEGs of road signs}

... }

REG = ALL?

Z-1)  $\exists x \in \text{ALL} . \quad x \notin \text{REG}$

$\pi$        $\uparrow$                            $\uparrow$   
language      all possible      languages defined  
(some problem)  
languages      by DFAs

option 1:  $\forall y \in \text{REG} . \quad x \neq y$

option 2:  $\forall y \in \text{REG} . \quad P(y)$   
 $\neg P(x)$

mystery #1: What is  $x$ ? witness

#2: What is  $P$ ? property

~~the~~

$\Rightarrow$

proof 1:  $\forall y \in \text{REG} . \quad P(y)$

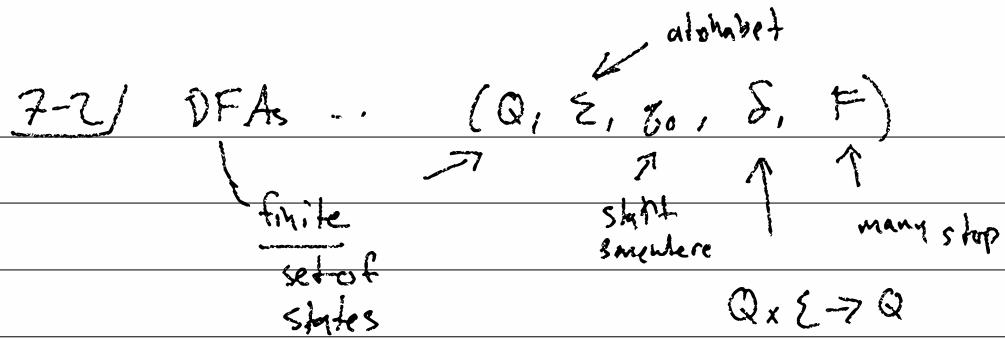
proof 2:  $\neg P(x)$

$\Rightarrow$

$x \in \text{ALL}$ , but  $x \notin \text{REG}$

$\Rightarrow$

computers aren't omnipotent



$\text{EQ}$

$$\epsilon \in \text{EQ}$$

$$01 \in \text{EQ}$$

$$0011 \in \text{EQ}$$

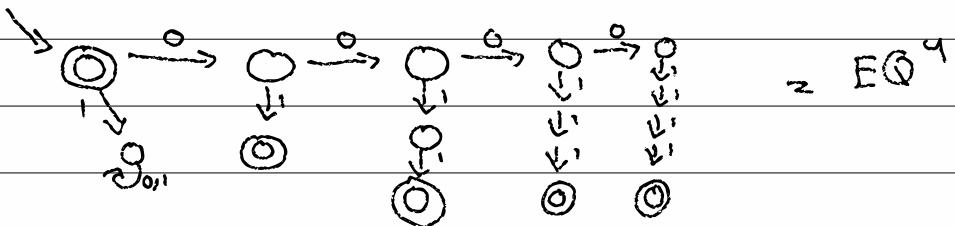
$$0 \notin \text{EQ}$$

$$010 \notin \text{EQ}$$

$$000111 \notin \text{EQ}$$

$$0110 \notin \text{EQ}$$

$$00001111 \notin \text{EQ}$$



$$|\text{EQ}^0| = 2$$

$$|\text{EQ}'| = 4 = |\text{EQ}| + 2$$

$$|\text{EQ}^3| = 11 = |\text{EQ}^2| + 4$$

$$|\text{EQ}^2| = 7 = |\text{EQ}| + 3$$

$$|\text{EQ}^n| =$$

$$|\text{EQ}^{n-1}| + n \cdot 2$$

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

$$\forall n. 0^n 1^n \in \text{EQ}$$

$$\wedge 0^n 1^n \in \text{EQ}^m \text{ where } n \leq m$$

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

7-3/ int count = 0; char c;  
while (~~(c == '0')~~ c = getchar(); ~~(c == '0')~~)  
    count++;  
~~update(c);~~  
while (c = getchar()) c == '1'  
    count--);  
return count == 0;

⇒

EQ<sup>M</sup>, what is m?  $m = 2^{31} - 1$

$$m = 2^{2^{31}}$$

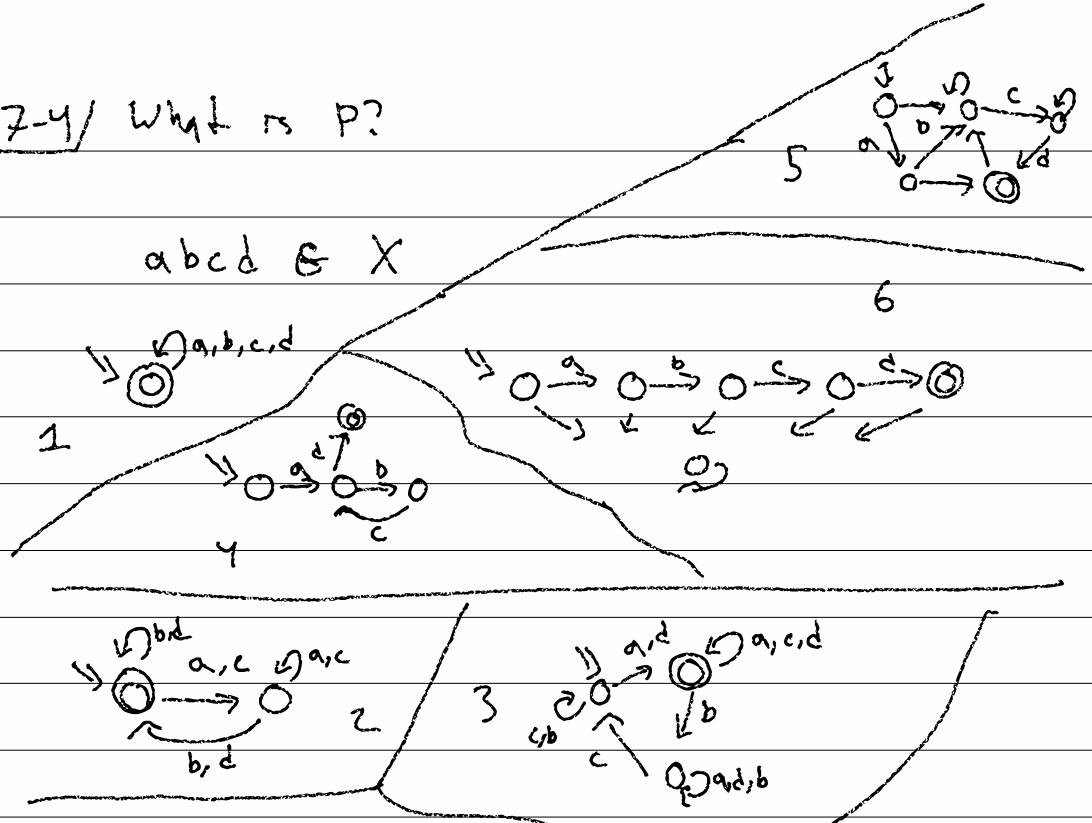
~~EQ<sup>2^31</sup>~~

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

~~X~~ =

Step 1: What is X? ✓

Z-Y / Why is P?



1:  $a a a a b c d \in X$

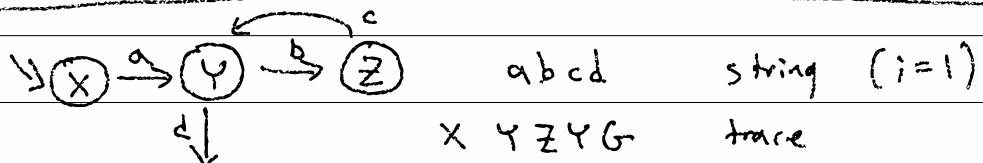
2:  $a b a b a b a b c b \in X$

5:  $\times$

3:  $a b c a b c a b c d \in X$

4:  $a b c b c b c d \in X$

6:  $\times$

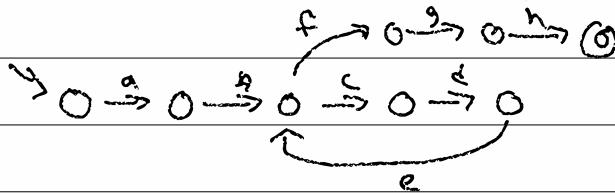


**5**       $a b c b c b c d$       string      ( $i=3$ )  
 $X Y Z Y Z Y G$       trace

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

$a d$       string      ( $i=0$ )  
 $X Y G$

7-5) DFAs must contain cycles!



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

ex:  $s = abcd$        $s$  visited 4 times       $|s|=4$

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

where  $x$  goes from  $q_0$  to  $q_r$

( $y$  is not empty  $\neq \epsilon$ )  $y$  goes from  $q_r$  to  $q_r$

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

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

That means for all:

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

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

## 7-6/ Regular Pumping Property (RPP)

RPP ( $A$  : Language) :=

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

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

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

$\wedge |y| > 0$ )

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

$xy^i z \in A$ .

Pumping Lemma:  $\forall A \in \text{REG}$ , RPP( $A$ ).

$p = |qi|$ ,  $x$  is the string before  $q_1$

$y$  is from  $q_1$  to  $q_2$

$z$  is from  $q_2$  to  $q_f \in F$

Step 2: What is  $p$ ? ✓

Step 3:  $\forall A \in \text{REG}$ , P( $A$ ). ✓

Step 4:  $\neg P(\text{EQ}) \dots$

8-1  $\neg \text{RPP}(\text{EQ})$

$\forall p \in N.$

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

$\forall (x, y, z \in S^*)$

$|y| > 0$

$|xy| < p$ )

$\exists i \in N$

$xy^iz \notin A$

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

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

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

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

$\neg \text{RPP}(\text{EQ})$

given  $p.$

choose  $s \in \text{EQ}$  where  $|s| > p$

$s = 0^p 1^p$

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

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

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

choose  $i$   $(i=0)$

$xy^iz \notin \text{EQ} \quad xy^iz = 0^a 0^{b+i} 0^c 1^p \notin \text{EQ}$

iff  $a+bi+c \geq p$

$a+bi+c \geq a+b+c$

$b_i \geq b$

$i \neq 1$

8-2/  $s = xyz \in A$  and  $xyz \in A$   
then

$xy^*z \in A$   
Regular expression

$xy^*z \subseteq A$

ALL  $\neq$  REG because  $EQ \in ALL$

$EQ \notin REG$

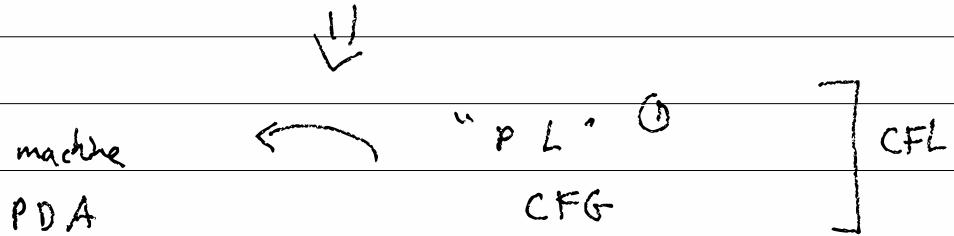
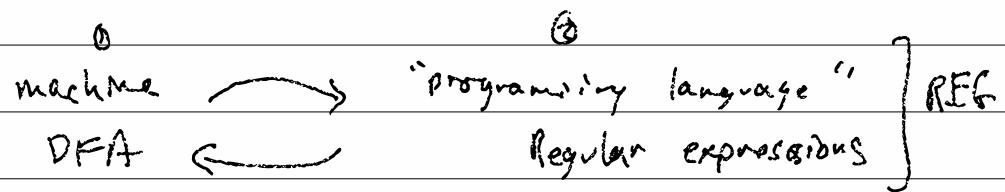
$$MEQ^0 = 0^0 \circ EQ$$

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

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

$$000 \circ 0011 \notin MEQ^3$$

9-1 EQ  $\ni 0^n 1^n$  for some  $n$



CFG : context-free grammar substitution

$S \rightarrow OS1$  LHS is always a production derivation

a CFS for  $0^n|n$        $V=SS3$       "variable" (non-homomols) / symbols

$S \Rightarrow S_1 \mid S_2$  RHS is a string of vars + terminals

$\Sigma = \{\epsilon, 1\}$  First var is "start symbol"

$E \rightarrow N$  | E O E

$$N \Rightarrow \emptyset$$

$0 \rightarrow '+' / '-' / 'x' / ' \div '$

q2/ membership       $x \in A$

defn  $A = \{ \dots, \cdot, \sim \}$

generation it produces  $s_1, s_2, s_3, \dots$

$S \rightarrow \varepsilon \quad | \quad OS1 \quad = g \quad \leftarrow$

$\overbrace{S \rightarrow OS1} \rightarrow OS11 \rightarrow 0011 \quad RL(g)$

$(S, [ \circ, \cdot, \rightarrow ], \{0, 1\}^G)$  = parse tree

$(S, [0,$   
 $(S, []),$   
 $1]),$

$$\text{PT} = \text{Var } x$$

$$(\Sigma + PT)^*$$

$\text{pt2str}(\text{V}, \text{seg}) = \text{seg2str seg}$

Seq2str [ ] = ε

Seg2str     $c \in \Sigma :: seg = c :: seg2str\ seg$

seq 2 str ~~PT~~ = pt 2 str PT +  
<sub>o</sub> seq 2 str seq

CFG  $g = (V, \Sigma, R, S)$

↓  
↓  
↓  
 $v \in V$

9-3) ~~Augmenting grammar~~

$R : V \rightarrow \text{Set of } (V \cup \Sigma)^*$        $P(V \times (V \cup \Sigma)^*)$

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

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

$$(V \cup \Sigma)^* = \emptyset, 0, 1, \$, 00, 01, 0\$,$$

$$10, 11, 1\$ , \$0, \$1, \$\$ , \dots$$

$$V \times (V \cup \Sigma)^* = \{(S, \emptyset), (S, 0), (S, 1), \dots\}$$

$$P(\quad) = \Sigma^* \{ (S, \emptyset), \{ (S, \emptyset),$$

$$\{ (S, 0), \{ (S, 0) \} \}$$

cfggen  $g = \text{helper } g \circ g, S$

helper  $g \circ v =$

let rules =  $g, R \circ v$

rhs = random rules

return ( $V$ , map over  $s$  in rhs:

if  $s \in V$ , helper  $g s$

o.w.  $s$ )

q-y/ all-n-deep g n = helper g n g.S

all-n-deep 0^n 1^n 2 = ε, 01, 0011

helper g n v = if n=0, don't return, o.w.

let rules = g, R v

for rhs ← rules; do

→ return (V, map s ← rhs; do

iterator if s ∈ V, helper g (n-1) s  
o.w. s )

S → 0 S 1 → 00 S 1 1

REG = a language defined by some DFA

CFL - context-free languages = a lang defined by a CFF

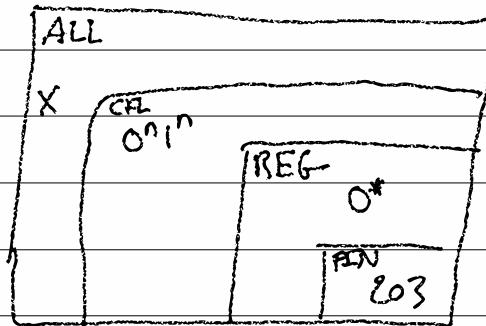
V S → S O V

S → N P | A N P | A A N P |

NP → N | PN

V → ... | Other V

Q-5)



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

$0^n * 0^n \in \text{CFL}$

$\text{REG} \subseteq \text{CFL}$

$X \subseteq Y$  iff  $\forall a \in X, a \in Y$

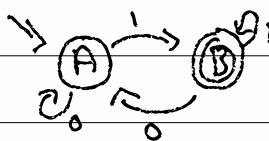
$a \in \text{REG}$  iff ~~there exists~~  $\exists d \in \text{DFA}, L(d) = a$

$a \in \text{CFL}$  iff  $\exists g \in \text{CFG}, L(g) = a$

$\underbrace{\forall d \in \text{DFA}, \exists g \in \text{CFG}, L(g) = L(d)}$

$\underbrace{V \cup}_{\text{args}}$   $\underbrace{E \cup}_{\text{result}}$  is a fun

args result



①	②
$A \Rightarrow 1B$	$0A$
$B \Rightarrow \epsilon$	$1B$

$$A \xrightarrow{1} B \xrightarrow{0} 11B \xrightarrow{1} 110A \xrightarrow{0} 1100A \xrightarrow{1} 11001B \xrightarrow{0} 11001$$

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

out: CFG =  $(V, \Sigma, R, S)$

$$V = Q \quad \Sigma = \Sigma \quad S = q_0$$

If  $\delta(q_i, c) = q_j$ , then  $R \ni q_i \rightarrow cq_j$

If  $q_i \in F$ , then  $R \ni q_i \rightarrow \epsilon$

7-6/ dfa-accepts d

(pt2str (cfggen (dfa2cfg d))) = true

10-1/ regular operations:  $\cup, \cap, \circ, *, -$

context-free ops:  $\cup, \circ, *$

$$g_1 \cup g_2 = S \Rightarrow (S_1) \mid (S_2)$$

Diagram illustrating the union operation ( $\cup$ ):

The expression  $g_1 \cup g_2$  is shown as  $S \Rightarrow (S_1) \mid (S_2)$ . This means that the start symbol  $S$  can derive either the string generated by rule  $S_1$  or the string generated by rule  $S_2$ .

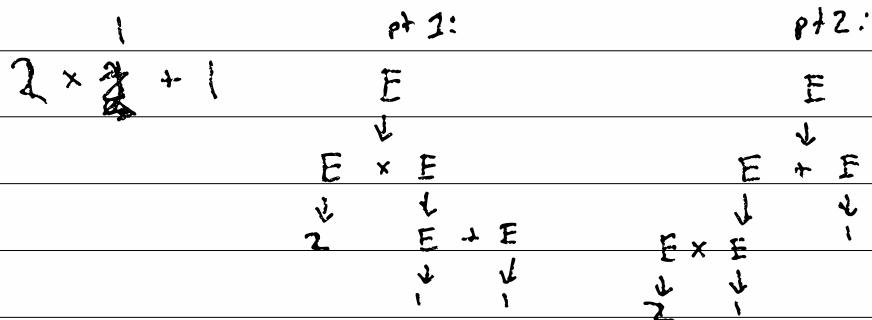
Rule  $S_1$  is defined by the production  $S_1 \rightarrow \dots$ , and its terminals are  $V_1 \rightarrow \dots$  and  $V_2 \rightarrow \dots$ . Rule  $S_2$  is defined by the production  $S_2 \rightarrow \dots$ , and its terminals are  $U_1 \rightarrow \dots$  and  $U_2 \rightarrow \dots$ .

$$g_1 \circ g_2 = S \Rightarrow S_1 S_2$$

$$g_1 * = S \Rightarrow \epsilon \mid S, S$$

10-2  $s \in L(g)$  iff  $\exists p \in g \text{ str}(p) = s$

$$E \hookrightarrow E \times E \quad | \quad E + E \quad | \quad 1 \quad | \quad 2$$



Ambiguous =  $\exists s. \cancel{s \in \text{pt}_1} \vee \text{pt}_1, \text{pt}_2, \text{str}(\text{pt}_1) = s$

$\wedge \text{str}(\text{pt}_2) = s \not\rightarrow \text{st}_2$

$$\begin{array}{l|l|l} E \Rightarrow E + E & \cancel{E \Rightarrow E} F & ] \text{ unambiguous} \\ F \Rightarrow F \times F & 1 & 2 \end{array}$$

ambiguous? : CFG  $\Rightarrow$  Bool

deambiguate :  $CFG \Rightarrow CFG$  s.t. amb? = #false

LL(k)

LHR

LR

10-3/  $V \Rightarrow a b F c d G + 1 e V$  //complex

$$x+y = y+x$$

$$0+y = y = y+0 \quad \checkmark$$

$$(1+n)+y = 1 + (n+y) = (n+y)+1 = (y+n)+1 \quad \checkmark$$

$$\text{assume } n+y = y+n \quad = y+(n+1)$$

CFG == NFA (Naam)

$\Updownarrow$  CNF = Chomsky Normal Form

CNF == DFA

A grammar  $g$  is in CNF iff

If  $r \in R$ , then  $r = A \rightarrow BC$  where

or  $r = S \rightarrow \epsilon \quad B \in V, C \in V$

or  $r = A \rightarrow a \quad \text{and } B \neq S, C \neq S, a \in \Sigma$

10-y /  $S \Rightarrow ASA \mid aB$

$A \Rightarrow B \mid S$

$B \Rightarrow b \mid \epsilon$

$S' \Rightarrow S$

Add a new start sym

$S \Rightarrow ASA \mid aB$

$A \Rightarrow B \mid S$

$B \Rightarrow b \mid \epsilon$

Remove all  $\epsilon$ s

$S' \Rightarrow S$

( $V \rightarrow \epsilon$ )

$S \Rightarrow ASA \mid aB \mid a$

$A \Rightarrow B \mid \epsilon \mid S$

$B \Rightarrow b$

$S' \Rightarrow S$

$S \Rightarrow ASA \mid SA \mid AS \mid S \mid aB \mid a$

$A \Rightarrow B \mid S$

$B \Rightarrow b$

Remove unit rules

$S' \Rightarrow ASA \mid SA \mid AS \mid aB \mid a$

( $V \rightarrow U$ )

$S \Rightarrow ASA \mid SA \mid AS \mid aB \mid a$

$A \Rightarrow b \mid ASA \mid SA \mid AS \mid aB \mid a$

$B \Rightarrow b$

Add intermediate vars

$S' \Rightarrow XA \mid SA \mid AS \mid 4B \mid a$

$X \Rightarrow AS$

$S \Rightarrow XA \mid SA \mid AS \mid 4B \mid a$

$Y \Rightarrow a$

$A \Rightarrow b \mid XA \mid SA \mid AS \mid 4B \mid a$

$B \Rightarrow b$

$$\frac{10-5/ \quad S \Rightarrow \epsilon \mid OS1}{S' \Rightarrow S} \text{ add } S'$$

$$\frac{S \Rightarrow \epsilon \mid OS1}{S' \Rightarrow S \mid \epsilon} \text{ removed } V \Rightarrow \epsilon$$

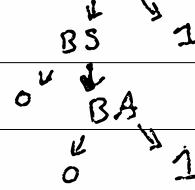
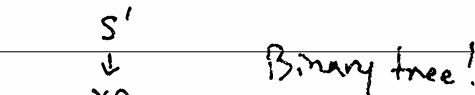
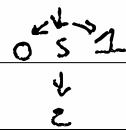
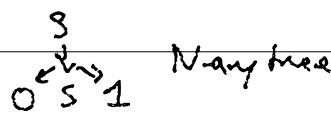
$$\frac{S \Rightarrow OS1 \mid 01}{S' \Rightarrow OS1 \mid 01 \mid \epsilon} \text{ remove } V \Rightarrow A$$

$$\frac{S \Rightarrow OS1 \mid 01}{S' \Rightarrow XA \mid BA \mid \epsilon} \text{ add mItem}$$

$$S \Rightarrow XA \mid BA$$

$$A \Rightarrow 1 \quad B \Rightarrow 0 \quad X \Rightarrow BS$$

$$S' \Rightarrow XA \Rightarrow BSA \Rightarrow OS1 \Rightarrow OXAA \Rightarrow OBSAA \Rightarrow OSAA \Rightarrow \\ OOBAAA \Rightarrow OOOAAA \Rightarrow OOO1AA \Rightarrow OOO11A \Rightarrow OOO111$$



CNF parse  
trees are  
binary!