

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 Σ

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

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

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

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 Σ

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

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

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

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 ϵ often

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

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

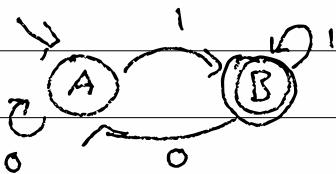
The set of strings in the lexicographic ordering

of $\Sigma \approx \Sigma^*$

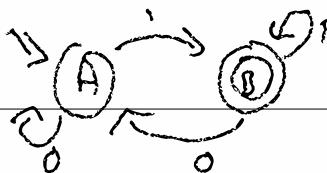
$\epsilon \in \Sigma^* \quad \text{PPCBSPPPP} \in \Sigma^*$

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	B
0	A A
1	B B

always finite
are the states

$\{\epsilon, A, B\}$

$\{\epsilon, 0, 1\}$

startstate
= A

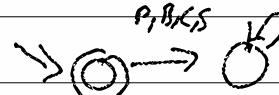
accepting states
 $\{\epsilon, B\}$

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

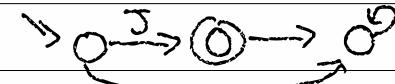
No string DFA:



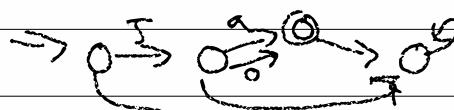
only empty string:



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

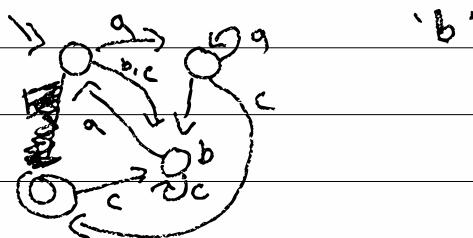
Transducers are DFAs where state writers
Moore machines

$L(d)$ = 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

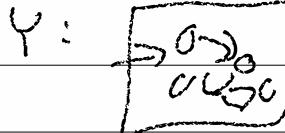
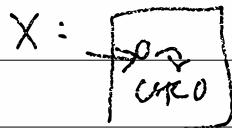
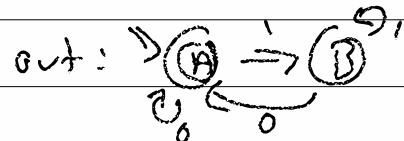
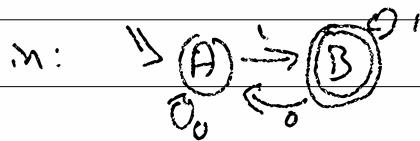
$$L(d') = \overline{L(d)} \quad (\text{ie } d' \text{ says } s \text{ yes})$$

negate: DFA \rightarrow DFA

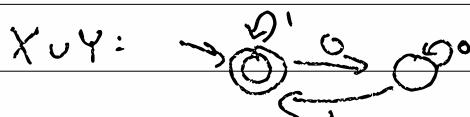
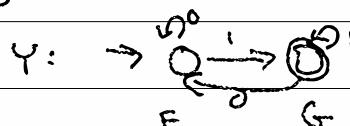
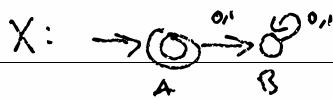
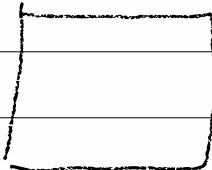
when d says no

negate (tddcs) = Evens

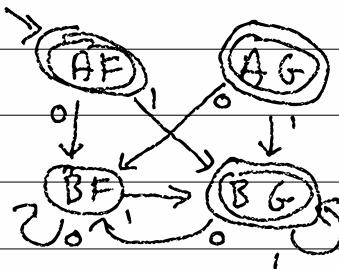
& vice versa)



$X \cup Y$:



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

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

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

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

$$X \subseteq Y$$

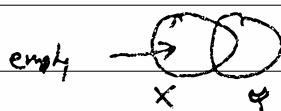
$$\text{and } Y \subseteq X$$

subset? : DFA \times DFA \rightarrow bool

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

(epsilon) (Even) = Yes

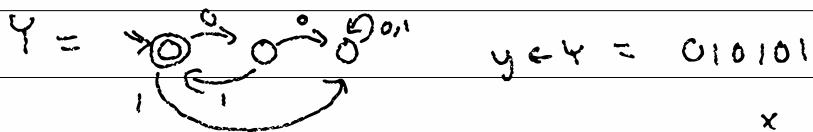
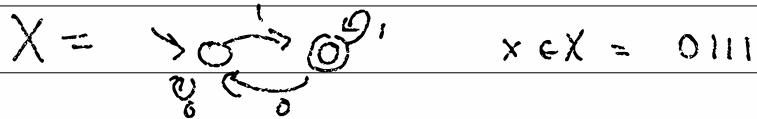
(epsilon) (odd) = No



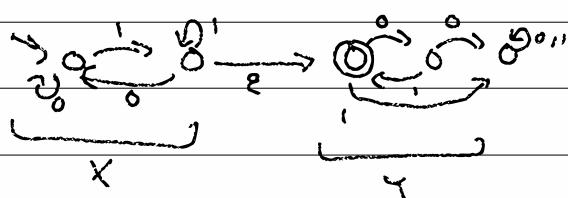
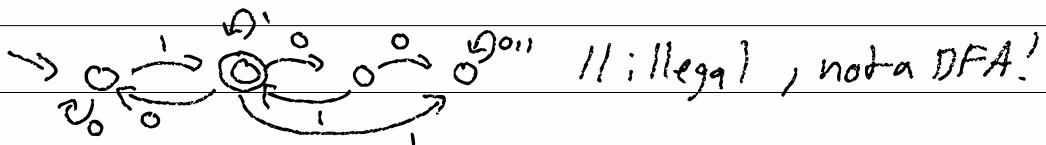
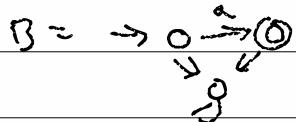
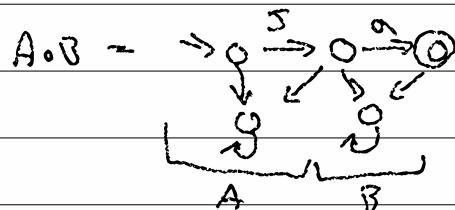
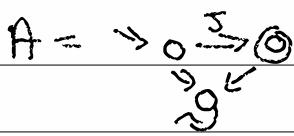
$X - Y$ must be empty

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

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



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

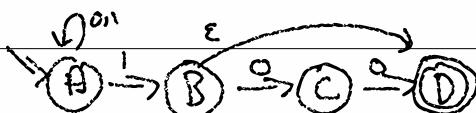


$F \xrightarrow{\quad} G$
 (skip from F to G
 at the right
 time)

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	εC3	εD3	ε3
1	εAB3	ε3	ε3	ε3
ε	ε3	εD3	ε3	ε3

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

$x \notin L(n)$ 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

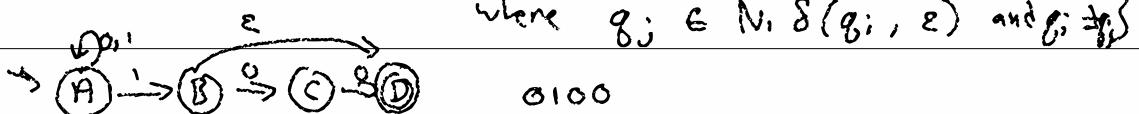
helper N g_i s =

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

$c :: s' \rightarrow$

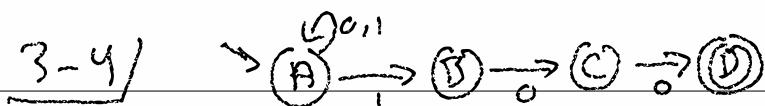
$\{ \text{++} \mid ++ = \text{helper } N g_j s'$
 where $g_j \in N, \delta(g_j, c) \}$

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



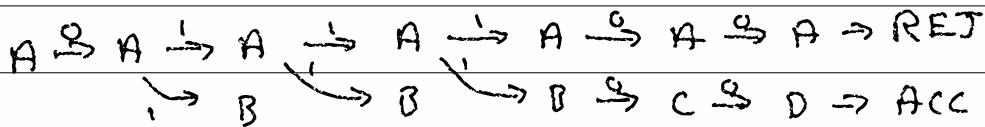
$0 \quad 1 \xrightarrow{A} \xrightarrow{B} \xrightarrow{C} \xrightarrow{D} \text{Reject}$

$A \xrightarrow{A} \xleftarrow{A} \xrightarrow{B} \xrightarrow{C} \xrightarrow{D} \text{ACCEPT}$
 $\epsilon \xrightarrow{D} \quad = (\text{branch } D \text{ (accept)})$
 $= (\text{branch } D \text{ } \emptyset)$



"ends in 100"

011100



backtrack : NFA \times String Σ^* \rightarrow Bool

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

helper N g; S =

OR (case s with $[] \rightarrow g_i \in N.F$)

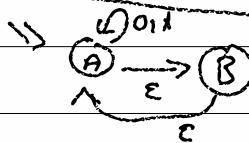
$c::s' \rightarrow OR_{g_j \in N.S(g_i, c)} helper(N.g_j.s')$

OR

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

helper N g; s

$x=3$; $(1 \parallel x++)$; $x==3$,



maybe DS = tree

unfold : $A \rightarrow DS(B)$

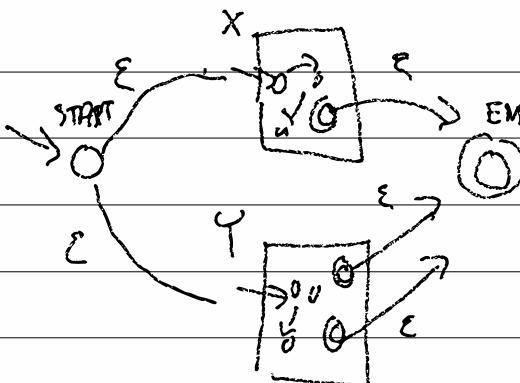
$$\text{fold} : \text{DS}(B) \rightarrow C$$

Haskell

there always exist

combined : A \Rightarrow C

4-1/



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

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

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

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

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

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

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

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

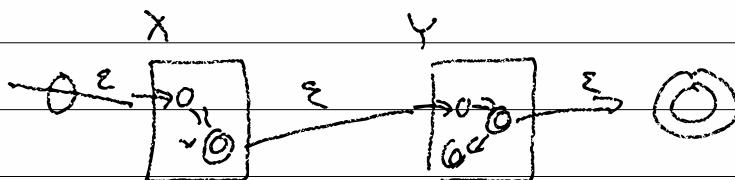
$$(\text{START}, \epsilon) = \{(g_{0x}, 0), (g_{0y}, 1)\} \quad \cup Q_Y \times \{\emptyset\}$$

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

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

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

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



4-2) Kleene-star X^*

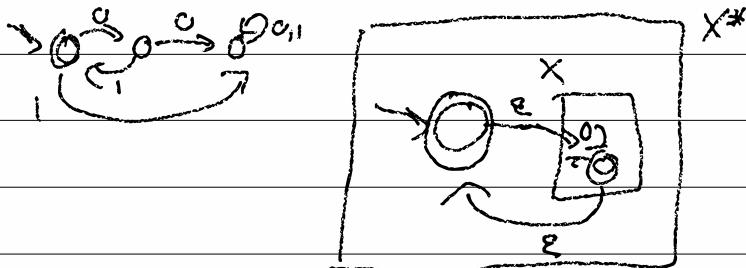
$\bar{z} \in X^*$ iff $\bar{z} = z$ OR $\bar{z} = xy$

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

iff $z = x_0 \dots x_n$

where $x_i \in X$

$(\{0\}^3 \{1\})^*$ = any number of 01 sequences

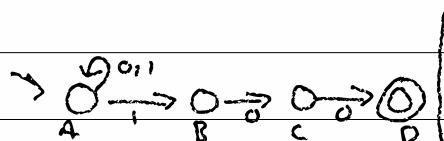


DFA's = $\cup, \cap,$ —

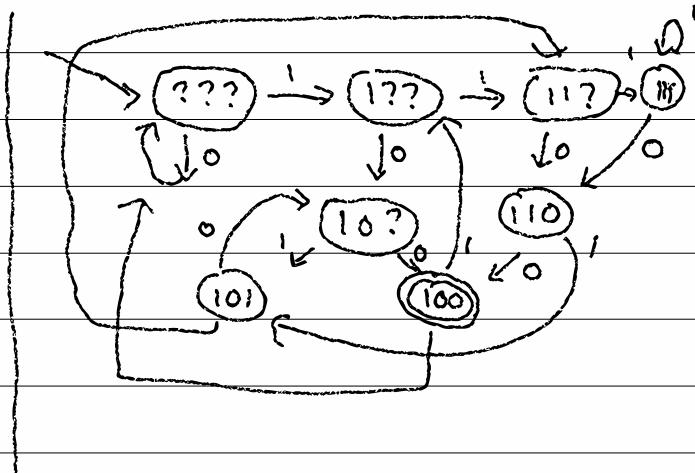
DEA \rightarrow NFA

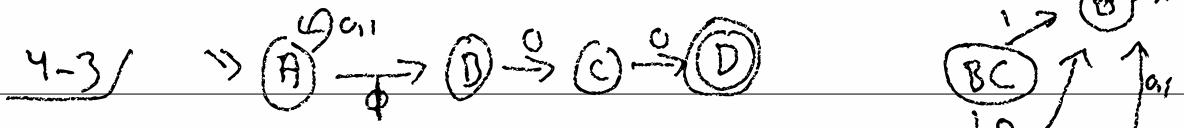
NFAs : \cup , \circ , *

wayt: NFA \Rightarrow DFA

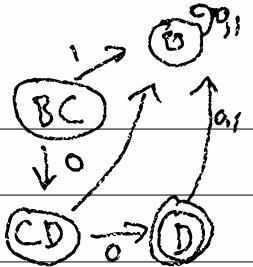


0100





01100
 $A \xrightarrow{0} A \xrightarrow{1} A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{0} A$
 $B \xrightarrow{0} B \quad C \quad D \leftarrow \checkmark$



???

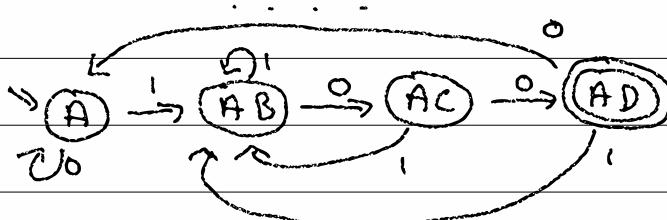
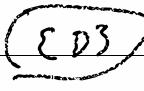
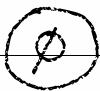
???

???

110?

110?

100



01100 ✓

011001 X

1111 X

111100 ✓

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

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

$$Q_D = P(Q_N) \quad \delta_D = \{ \delta_{q_n} \mid q_n \in Q_N \} \quad F_D = \{ q_n \mid q_n \in F_N \}$$

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

~~q_n ∈ q_D~~ ~~ε~~ ~~ε~~

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

Q_M

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

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

random string

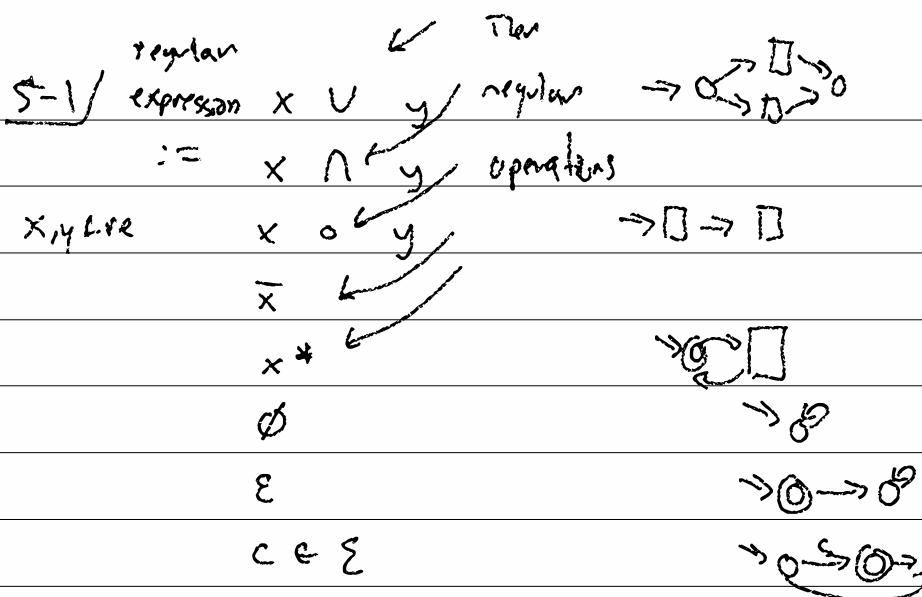
= pick a random number

(lex; n)

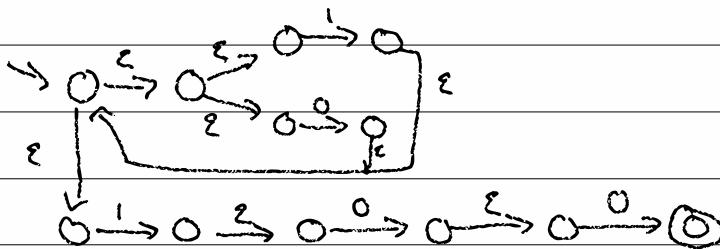
write the NFA manually as a VKA...

use the dfa equality checker to see
that

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



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



=

$$\rightarrow Q \xrightarrow{1} Q \xrightarrow{0} Q \xrightarrow{0} Q \xrightarrow{\varepsilon} Q \xrightarrow{0} Q \xrightarrow{0} F$$

$$* . c \Rightarrow \varepsilon^* . 1 . 0 . 1 . c'$$

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 ϵ = ϵ

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 ϵ pr = pr ϵ

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_s \leftrightarrow NFA_s



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

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: $0 \xrightarrow{\delta} 0$

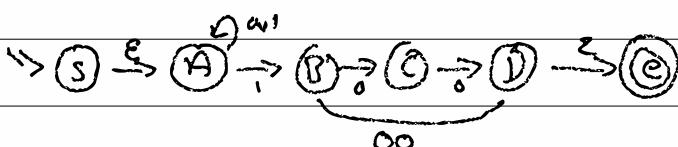
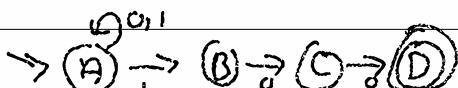
GNFA (generalized NFA)

 $0 \xrightarrow{\delta} 0$ $= (Q, \Sigma, g_a, \Delta, g_f)$ 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 

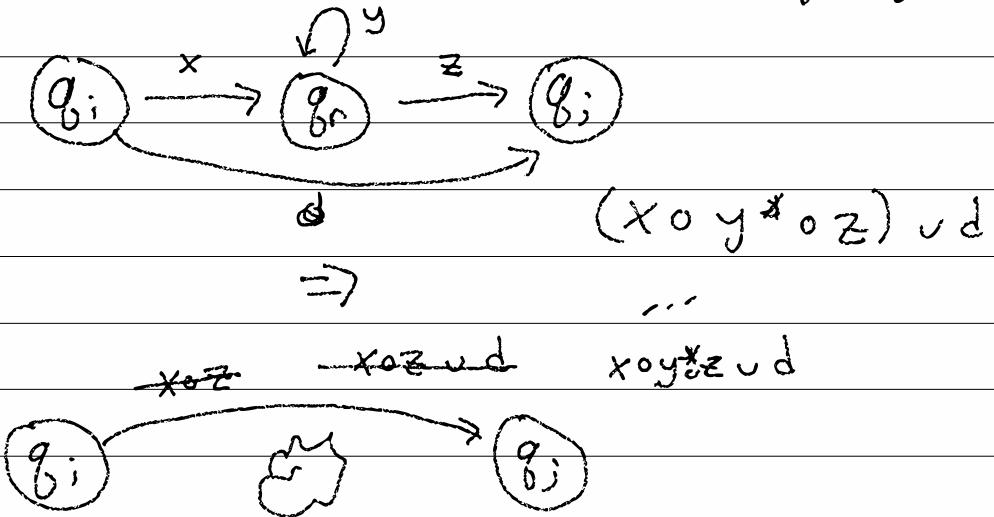
if

 $\Delta(g_i, g_j) = r \quad \cancel{x \in r} \quad x \in r \text{ then } g_i \xrightarrow{x} g_j \text{ in the NFA}$ 

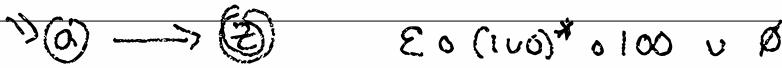
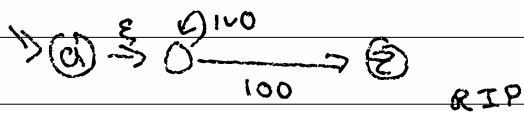
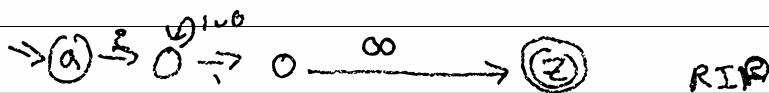
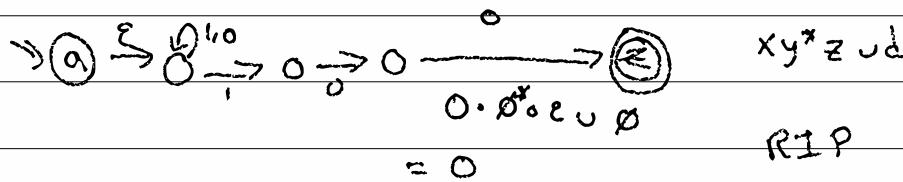
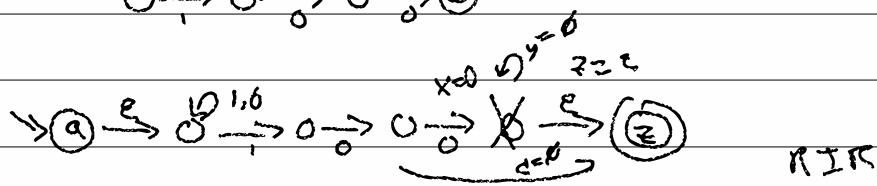
6-2/ $R_{10} : \text{GNFA } (n+1) \rightarrow \text{GNFA } (n)$

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

q_r is gone



IN

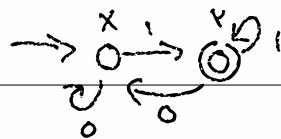


$$E_0 (1 \cup 0)^* \cdot 100 \cup \emptyset$$

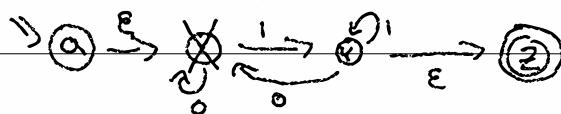
$$= (1 \cup 0)^* \cdot 100 \quad \leftarrow$$

OUT

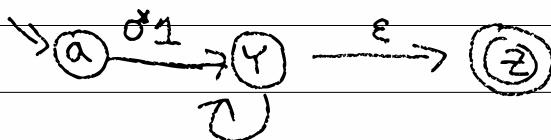
6-4)



$\Downarrow \text{IN}$



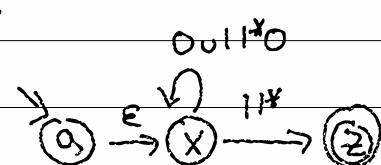
$\Downarrow \text{RIP}$



$1 \cup 00^*1$

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

//



$\Downarrow \text{RIP}$

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

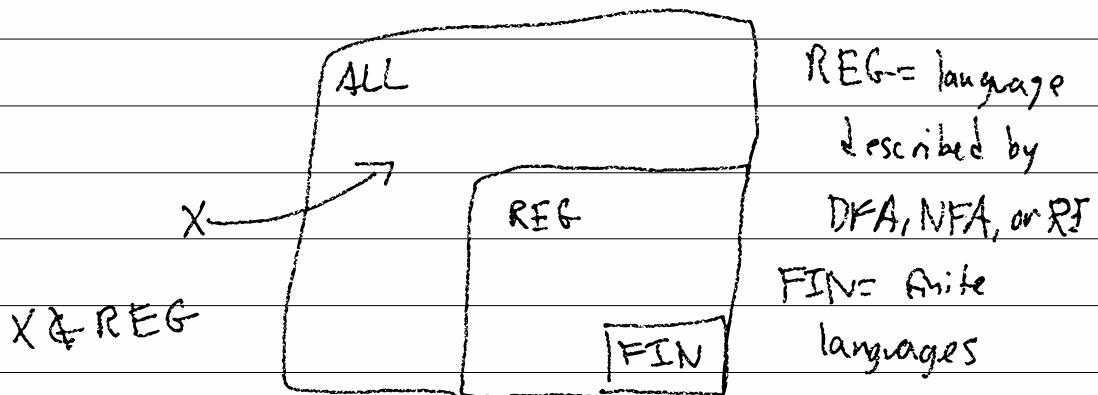
$(01)^*1$

DFA $\xleftarrow{\quad} \text{REX}$

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

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?