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

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

which statements are true?

"All birds have wings"

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

Defining the set of true stmts

Making a decision procedure

Generating a list

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

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

A string of Σ is a finite sequence of Σs

PPPP CSCSCSSB $\quad = \epsilon$

1-3/ A language is a set of strings

{ ε, P, PP, PPP, PPPP } – finite

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

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

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

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

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

→ complement or negation of Y

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

$PP \circ BC = PPBC$

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

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

$PPC \in$

is a sequence

lexicographic ordering of Σ

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

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

...

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

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

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

lexi 0 = ε lexi 1 = 0 lexi 2 = 1

lexi 3 = 00

lexi n = $\quad sz$ = size of Σ

if $n < sz^0$ then ret ε often!

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

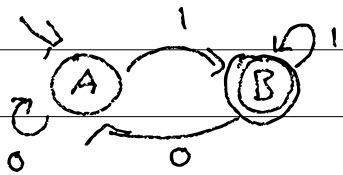
$(n-sz^0) - sz^1 < sz^2$ convert of len 2

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

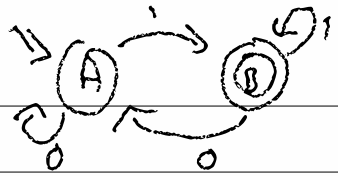
$\varepsilon \in \Sigma^*$ PRCBSPIPP $\in \Sigma^*$

0101111 $\in \Sigma^*$

Deterministic Finite Automata (DFA)



2-1) DFA



◎ means Yes

○ means No

0110 = No 1 = Yes

0111 = Yes 11 = Yes

0010 = No 00 = No

1101 = Yes

ε = No

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

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

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

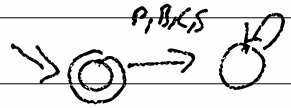
	A	B
0	A	A
1	B	B

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

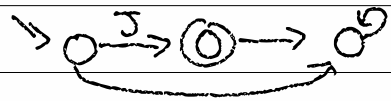
"n % 2 == 1" i.e. "odd? n"

No string DFA:

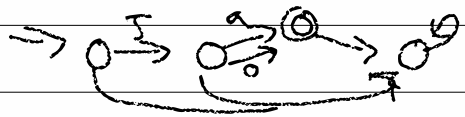
only empty string:



only the string 'J':



'Ja' and 'Jo'



$: Q \times \Sigma \to Q$

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

accepts? d s : DFA × Σ^* ⇒ Bool

accepts? d ε = is q_0 in F?

d.F.member (d.q_0)

accepts d (c :: s)

: DFA : Q : Σ^*

accepts d s = helper d d.q_0 s

helper d q_i ε = $q_i \in$ d.F

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

q_j = d.δ(q_i, c)

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

↙ trace

2-3/ 0110 ⇒ Even, Odd, Odd, Even

Transducers are DFAs where trace matters

Moore machines

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

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

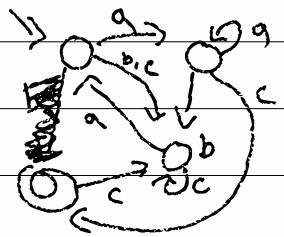
may be infinite

Given a DFA, return a string that would be accepted

example : DFA ⇒ Σ^* or false

s.t. if example d returns s then

accepts? $d\ s$ = true

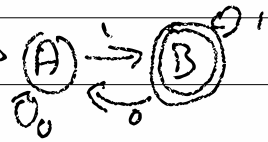


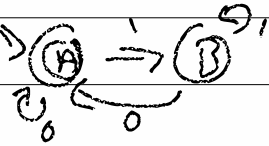
2-4/ Suppose had d in a DFA, construct d' where

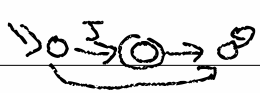
$L(d') = \overline{L(d)}$ (i.e. d' says yes when d says no & vice versa)

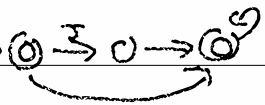
negate : DFA → DFA

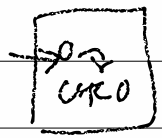
negate (Odds) = Evens

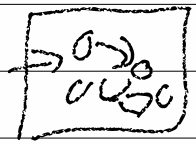
in: 

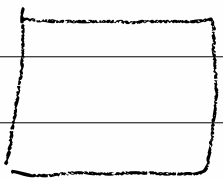
out: 

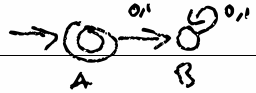
in: 

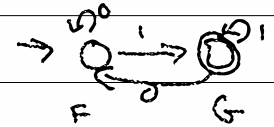
out: 

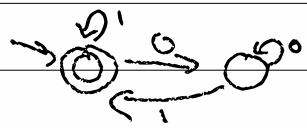
X: 

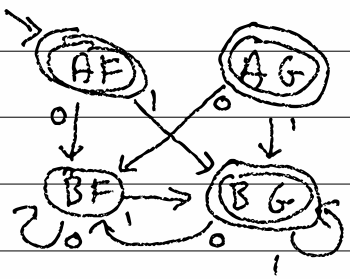
Y: 

X ∪ Y: 

X: 

Y: 

X ∪ Y: 

Z: 

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

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

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

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

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

and to make intersect

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

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

$X \subseteq Y$ (subset) iff

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

$X = Y$ iff

$X \subseteq Y$

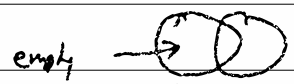
and $Y \subseteq X$

subset? : DFA × DFA ⇒ bool

subset? (≫ϕ) X = Yes

(epsilon) (Evens) = Yes

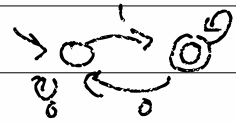
(epsilon) (odds) = No



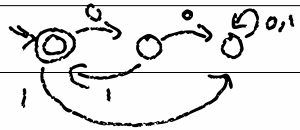
$X - Y$ must be empty

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

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

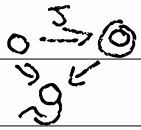
$X =$ 

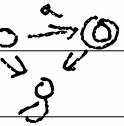
$x \in X = 0111$

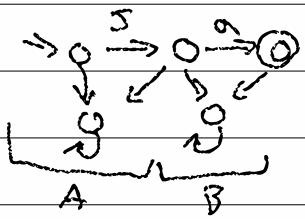
$Y =$ 

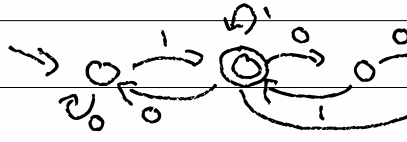
$y \in Y = 010101$

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

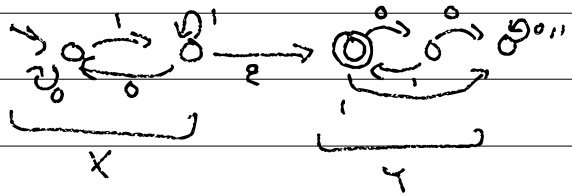
$A =$ 

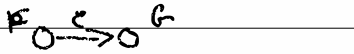
$B =$ 

$A \circ B =$ 



// illegal, not a DFA!





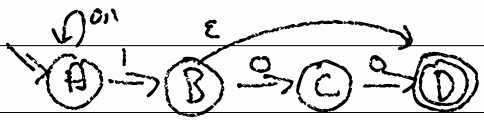
(skip from F to G at the right time)

3-2/ NFA - non-deterministic finite automata

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

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

(sigma, epsilon)



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

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

x ∉ L(n) iff

∀t. oracle n t = #f

(or "if")

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

(A,0) (B,1) (C,0) (D,0) 0100 ∈ L(n)

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

oracle interpretation : NFA x trace → boolean

oracle N t = helper N N.q0 t

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

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

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

o.w. "invalid trace"

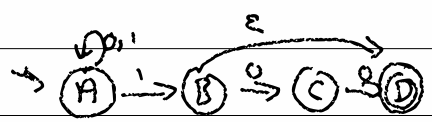
3-3/ trace-tree = accept | reject
| branch state (tt set)

all : NFA × Σ^* → tt

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

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

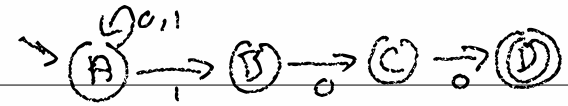
c :: s' →
{ tt | tt = helper N q_i s'
where $q_i \in N.\delta(q_i, c)$ }
∪ { tt | tt = helper N q_i s
where $q_i \in N.\delta(q_i, \varepsilon)$ and $q_i \neq q_i$ }



0100

0 1 A → A → A → Reject

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

3-4/ 

"ends in 100"

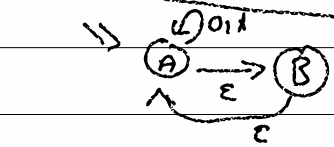
011100

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

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

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

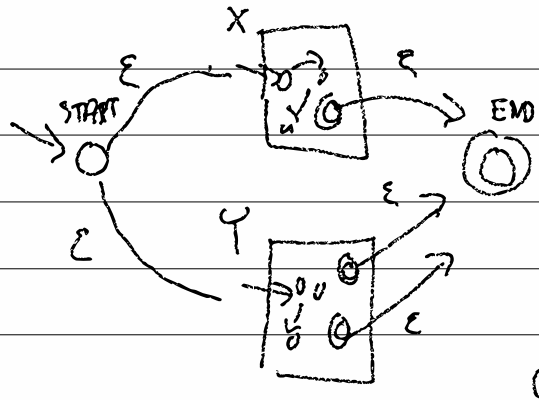
x=3 ; (1 || x++) ; x==3,



maybe DS = tree

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

4-1



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

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

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

$F_z = \{END\}$

$q_{0z} = \{START$

$Q_z = \{START, END\}$

$\delta_z(q_i, c) =$

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

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

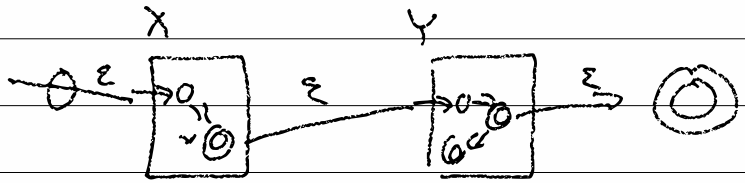
$(END, \_) = \{\}$

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

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

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

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



4-2/ Kleene-star X^*

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

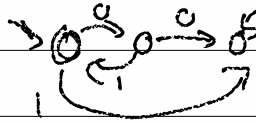
where $x \in X$ and

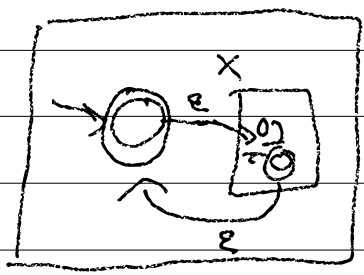
$y \in X^*$

iff $z = x_0 \dots x_n$

where $x_i \in X$

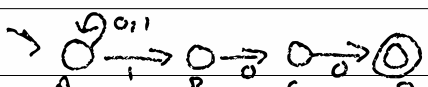
$(\{0\}\{1\})^*$ = any number of 01 sequences



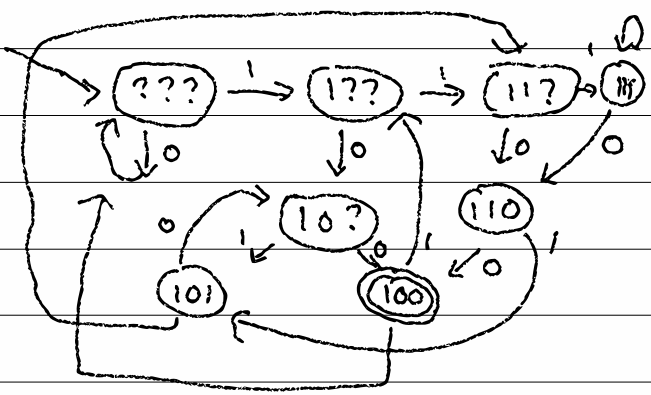


DFAs : $\cup$, $\cap$, $\overline{\ }$ DFA → NFA

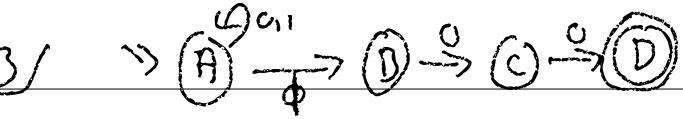
NFAs : $\cup$, $\circ$, $*$ want: NFA → DFA

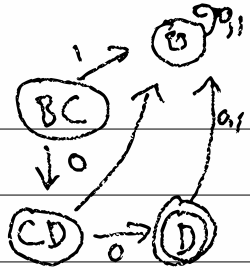


01100



4-3/



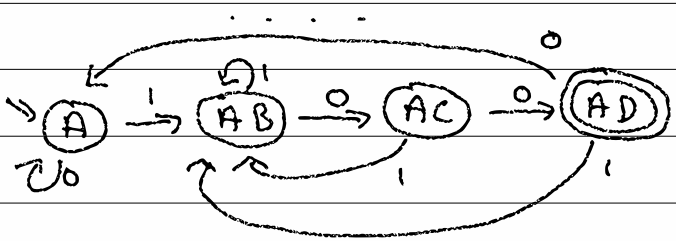


01100

A → A → A → A → A → A

B B C D ← ✓

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



01100 ✓

011001 X

1111 X

111100 ✓

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

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

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

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

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

4-4)

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

= accepts (NFA→DFA N) x

random string

= pick a random number

(lexi n)

write the NFA manually as a DFA...

use the dFA equality checker to see that

dfa-equal? manual (N→D N) = #t true