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

1-2] A set is "a bunch of stuff"

$\emptyset$ – nothin' in it

$\forall x, x \notin \emptyset$

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

$\{$ ✓ , ☺ $\}$

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

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

union – $\cup$ u u

$\forall x. \; x \in A \cup B$ iff $x \in A$ or $x \in B$

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

1-3) "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 $\emptyset$, $\{x\}$, $\cup$ are finite.

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

The <u>Universe</u> ($\mathcal{U}$)

subset

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

1-4 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

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

chars chars

$\overset{0}{\downarrow}\overset{1}{\downarrow}\overset{2}{\downarrow}$

$\underbrace{\text{"0100001"}}_{\text{length } = 7}$ = a string = s

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

$U = \Sigma^*$ ← special notation

$A^* = \{\varepsilon\} \cup \cancel{\ } A \circ A^*$

epsilon = "" = the string w/ no characters

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

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

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

L5/ #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())));
```

{Σ}

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

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

newList(Chars)
{ Zero, One};

0	1	2	3
ε	0,	00,	000
	1,	01,	001
		10,	010
		11,	011,
			100
			101
			110
			111
2^0	2^1	2^2	2^3

$|A|^i$ where i = layer I'm looking at

lexi : $\Sigma \times N \rightarrow \Sigma^*$

lexi B 0 = ε

lexi B 1 = 0

lexi B 2 = 1

lexi B 6 = $1\cancel{0}1$

2-1 "1+1" $\longrightarrow$ "2"

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

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

$\emptyset$ $\{x\}$ $A \cup B$

Alphabet Σ Universe Σ^*

$\{0, 1\}$ $\{\varepsilon,$ 0110, 000001,  $\}$

$\mathbb{P}(A)$ 2^A

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

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

$\emptyset \in P(A)$ $\emptyset \subseteq A$ 0110 $\{1,2\}$

$\{0\} \in$ " $\{2,3\} \in$ $\{0,2,3\}$ $\downarrow$ $\underline{0}12\underline{3}$

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

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

$\{\varepsilon\} \in P(\Sigma^*)$ 0011, ...

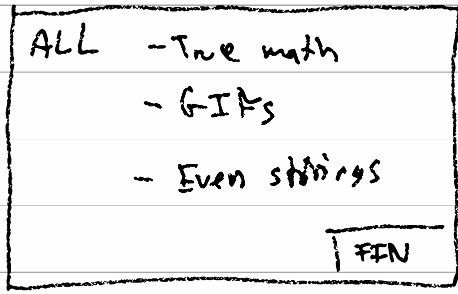
all even lengthed strings $\in P(\Sigma^*)$ $= \{\varepsilon, 00, 11, 01, \ldots\}$

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

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

2-2 ALL = $P(\Sigma^*)$

FIN = the set of finite sets

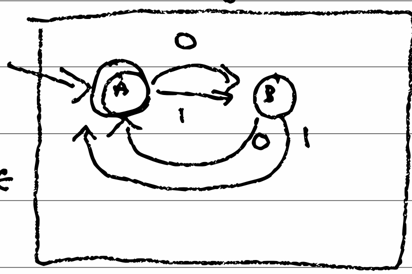


$\emptyset \in$ FIN

$\forall x \in \Sigma^*.\ \{x\} \in$ FIN

$A \in$ FIN $\wedge$ $B \in$ FIN $\rightarrow$ $A \cup B \in$ FIN

All even strings =



$x \in$

ε 01 ~~11~~

DFA - a deterministic finite automata

○ - states $\{A, B\}$

→□ - start state A

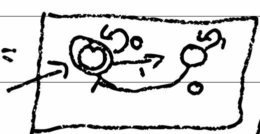
◎ - accepting states $\{A\}$

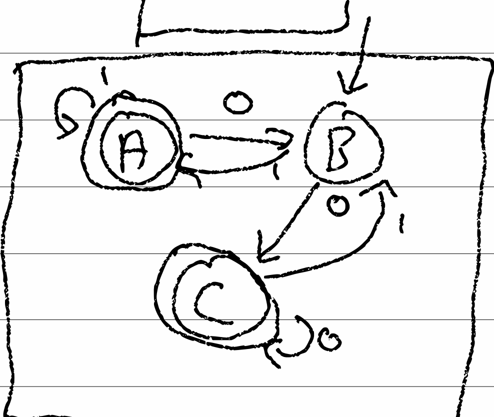
$\circ \xrightarrow{x} \circ$ - transition

x - labels are Σ

	0	1
A	B	B
B	A	A

even nums =





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

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

	0	1
A	B	A
B	C	A
C	C	B

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

DFA configuration $= Q \times \Sigma^*$
$[q]\ w$

config update function : config $\times$ DFA $\Rightarrow$ config
$[q]w \longrightarrow [q']w'$

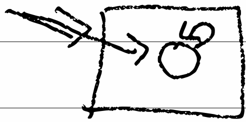
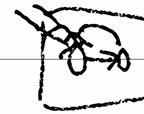
$[q_i]\ x y \Rightarrow [q_j]\ y$ iff $\delta(q_i, x) = q_j$

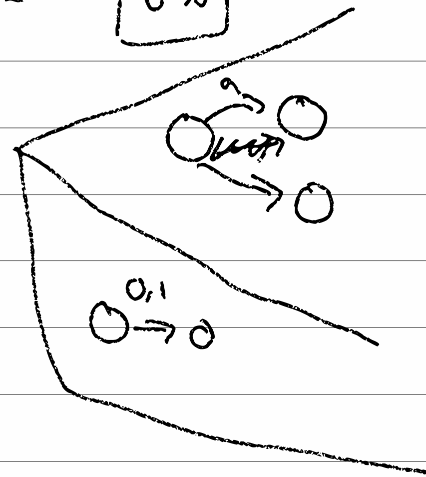
$x \in DFA$ iff $[q_0]x \Rightarrow \Rightarrow \Rightarrow \Rightarrow \Rightarrow [q_f]\ \varepsilon$
and $q_f \in F$

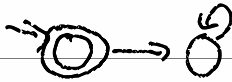
$0110 \in$ EvenLen iff $[A]0110 \Rightarrow [B]110 \Rightarrow [A]10$
$\Rightarrow [B]0 \Rightarrow [A]\ \varepsilon$ $A \in \{A\}$ ✓

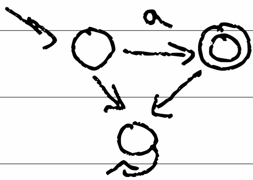
```
class DFA {
  ... Q, Σ, F, q0, δ ...
  public bool accepts (String x) {
    State qi = q0;
    while (!x.empty()) {
      qi = δ(qi, x.first());
      x = x.rest(); }
    return F.in(qi); } }
```

2-4

 $= \emptyset =$ 

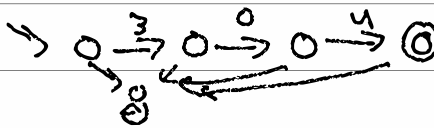


 $= \{\varepsilon\}$

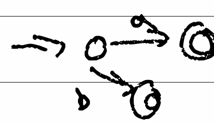


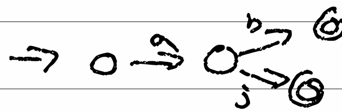
$= \{a\}$

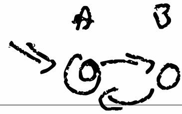
$=$ 

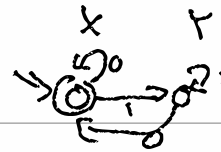
$\{304\} =$ 

$A \cup B \in$ DFA (if $A \in$ DFA and $B \in$ DFA)

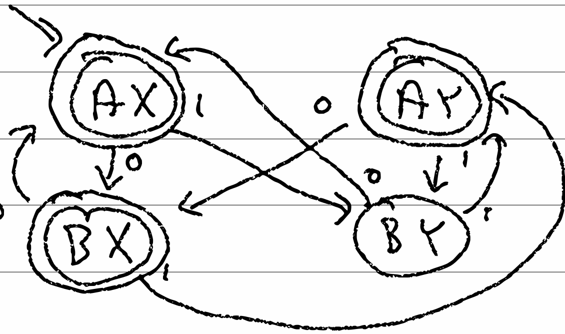
$\{a\} \cup \{b\}$ 

$\{a;\} \cup \{ab\}$ 

2-5| EvenLen 

IsEven 

$\cup$



ε ✓

00 ✓

11 ✓

0 ✓

110 ✓

$\cap$

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

iff $x \in A \wedge y \in B$

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

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

$X = A \cup B$

$Q_X = Q_A \times Q_B$

$q_{0X} = (q_{0a}, q_{0b})$

$F_X = F_A \times F_B$ — $\cap$

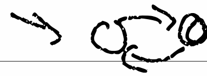
$F_A \times Q_B \cup Q_A \times F_B$ — $\cup$

$\delta_X((q_a, q_b), c) = (\delta_a(q_a, c), \delta_b(q_b, c))$

$x \in A \cap B$ iff $x \in A \wedge x \in B$

2-6/ $x \in A^c$ iff $x \notin A$ $(x \in U)$

Even Len odd len

 $\Rightarrow$ 

$F = \{A\}$ $F' = Q - F$

or F^c (wrt Q)

complement

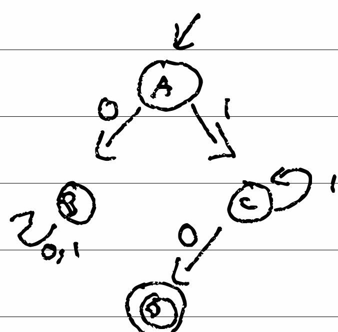
Algorithm for $X \subseteq Y$ if X & Y are DFAs

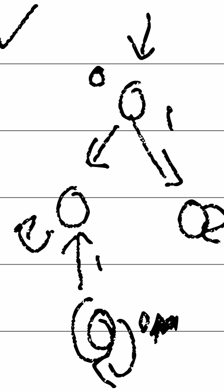
3-11 DFA ⇒ example or false

DFA:

- Q: State → Bool
- Σ: list of characters
- q_0: state
- δ: (State × Char) → State
- F: State → Bool

10 ✓





Q	$\emptyset$	q_0	
{A, B, C, D}	{}	[A]	A ⇒ ε
{B, C, D}	{A}	[B, C]	B ⇒ A, 0 C ⇒ A, 1
{C, D}	{A, B}	[C]	Yes, it's possible.'
{D}	{A, D, C}	[D]	D ⇒ C, 0
{}	{A, B, C, D}	[ ]	or if not, No!

10

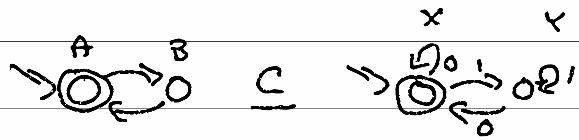
3-2/ subset

$\exists x \in U.\ x \in A \wedge x \notin B$

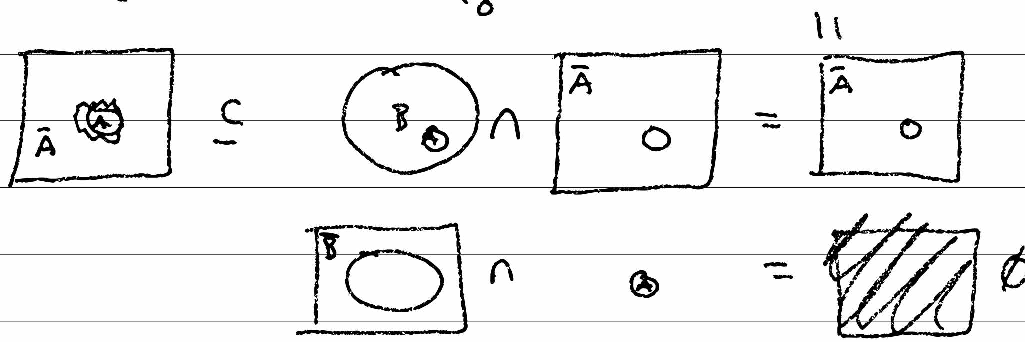
$A \subseteq B$ iff $\neg\ \forall x \in U.\ x \in A \Rightarrow x \in B$

$\{a, b\} \subseteq \{a, b, c\}$ $\quad U = \{a, b, c\}$

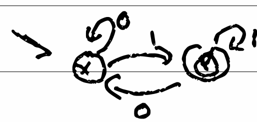
→ finite means naive works!

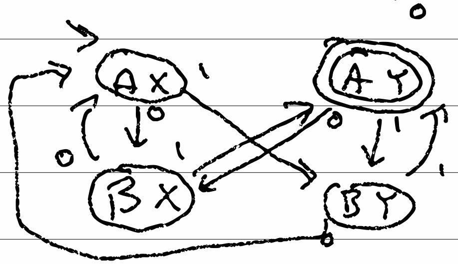


100



$\overline{\text{EvenNum}}$ =





→ 01

~~x~~ $\emptyset$

model checking

Soundness : model $\subseteq$ theory

completeness : theory $\subseteq$ model

model = theory

3-3/ 0, 1, 2, -1, 5 Z, P, N

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

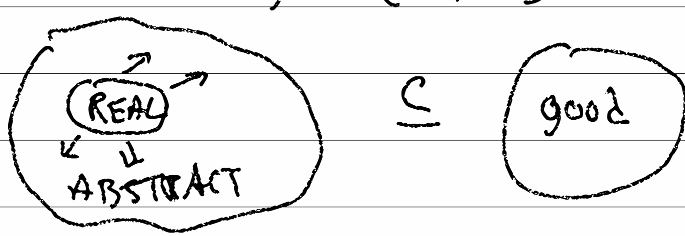
if $x > 0$ then

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

o.w

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

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



3-4/ Finite = $\emptyset \quad \{x\} \quad A \cup B$ $\quad \in$ DFA

$A^c \quad A \cap B \quad A \circ B$

Infinite = A^*

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

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

$\varepsilon \circ y = y$ if $a \in \Sigma$, $(a \circ x) \circ y = a \circ (x \circ y)$

$abcd = a \, b \circ c \, d$

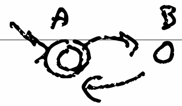
$\{j\} \circ \{m, n\} = \{jm, jn\}$

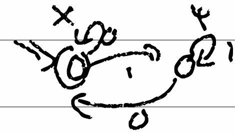
$x \in A^*$ iff $x = x_0 \circ x_1 \circ \ldots \circ x_n$ for $n \in \mathbb{N}$

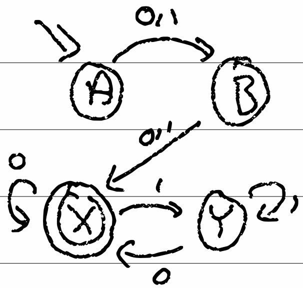
and $x_i \in A$

$\{jm\}^* \ni \varepsilon, \ jm, \ jmjmjmjm$

3-5/ Even Len $\circ$ Even Num



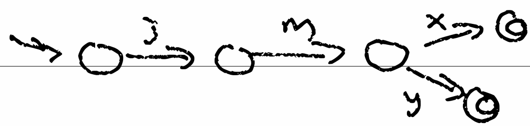


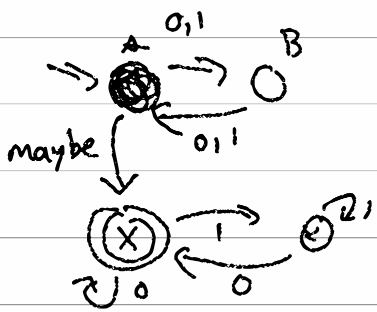


$\underbrace{00}\underbrace{110}$ ✓

0011 ×

$\{jm\} \circ \{x,y\}$

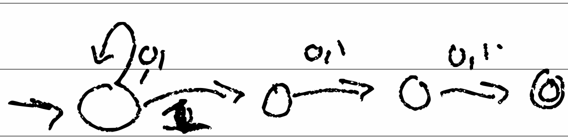




$\Sigma = \{0,1\}$

$x \in$ 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 a char?
- which path do you take?

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

$\delta'(A, 1) = \{A, B\}$

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

epsilon

$\varepsilon \quad \Sigma \quad e \quad \varepsilon$

4-1

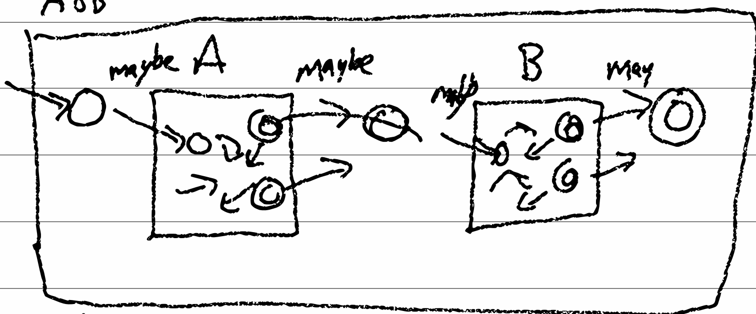
$A \circ B \in DFA$ iff $A \in DFA$

A^* $\quad$ $B \in DFA$

NFA (N - non D- deterministic)

$\delta : Q \times (\Sigma \cup \{maybe\}) \to P(Q)$ $\qquad$ $\delta : Q \times \Sigma \to Q$

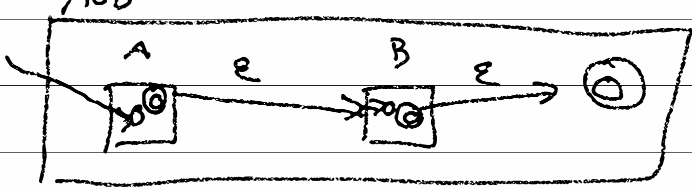
A∘B



maybe written as "ε"

What does (NFA) this mean?

A∘B



NFA ⇔ DFA

4-2 | What do NFAs mean?

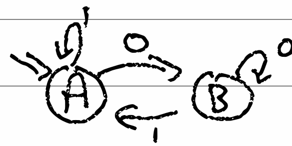
A DFA represents a set and
a set "is" a membership function

$U \Rightarrow \text{Bool}$

$= \Sigma^* \Rightarrow \{Y, N\}$

config $= \Sigma^* \times Q$

$\Sigma^* \Rightarrow Q^*$



$0110 \Rightarrow \underbrace{ABAAB}$ → a trace

$\Sigma^* \Rightarrow (\Sigma, Q)^*$

Σ_e ~~make~~ $= \Sigma \cup \{\varepsilon\}$

$0110 \Rightarrow \underbrace{(0,B)\,(1,A)\,(1,A)\,(0,B)}$

0A 1A 1A 0B ⇒ N

valid? : $Q \times (\boxed{\Sigma}, Q)^* \Rightarrow \{Y, N\}$ — DFA

valid q_i ε = Y

valid q_i (c, q_j) : more = if $\delta(q_i, c) \boxed{==} q_j$
valid q_j more
o.w. N

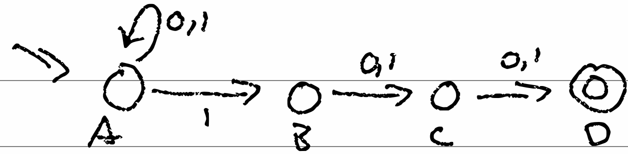
Nvalid? : $Q \times (\boxed{\Sigma_e} \times Q)^* \Rightarrow B$

Nvalid q_i ε = Y

Nvalid q_i (c, q_j) : more =
if $\boxed{q_j \boxed{\in} \delta(q_i, c)}$ then Nvalid q_j more
o.w. N

An Oracle

4-3



"3rd letter from end is 1"

0100   111   110100 — Y

000   1000   1011 — N

(0,A)(1,A)(0,A)(0,A) ✓ $str((\Sigma \times Q)^*) = \Sigma^*$

(0,A)(1,B)(0,C)(0,D) ✓ $str\ \varepsilon = \varepsilon$

$\underbrace{(0,B)}_{\delta(A,0)=\{A\}}(1,C)(1,D)\underbrace{(0,D)}_{\delta(D,0)=\emptyset}$ X   str (c,_)::move = c

c ∘ str move

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

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

valid q_0 t = Y

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

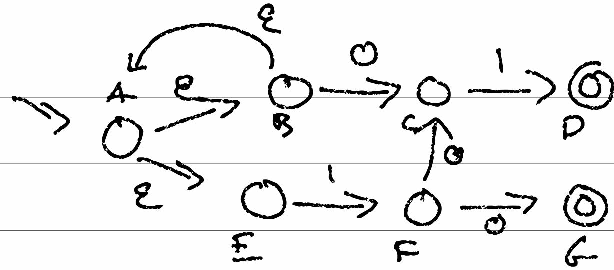
NFA-accepts : $\Sigma^* \to$ Y/N

figure all possible traces

check if valid and is strings match

check if last is in $\in F$

4-4



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

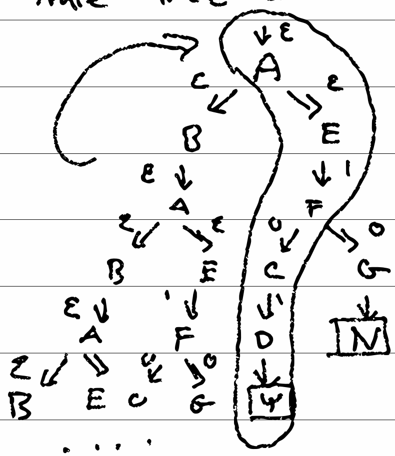
$(\varepsilon, E)\,(1, F)\,(0, C)\,(1, D)$ $\quad$ $101 = \varepsilon 010001$

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

$(\varepsilon, B)\,(\varepsilon, A)\ x$ where x is valid

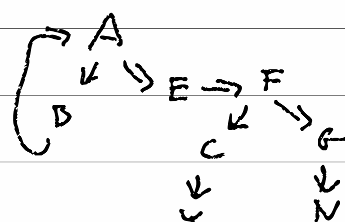
$\Rightarrow$ valid

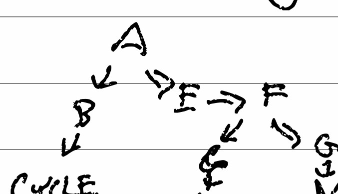
Trace Tree = Y | N | Branch (Σ, Q) (List TT)



101

ε, E $\quad$ $1, F$ $\quad$ $0C$ $\quad$ $1D$



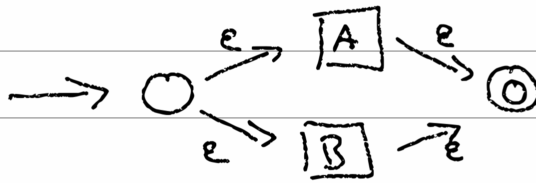


Forking model of NFAs (make TT)

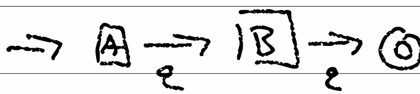
Back-tracking model (explores TT)

4-5/

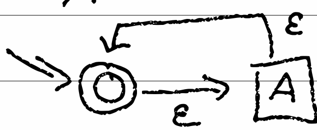
$A \cup B$

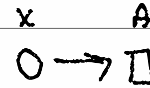


$A \circ B$



A^*





state X transitions to THE start state of A

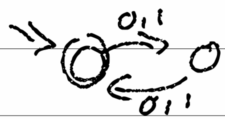
All accepting states of A transition to X

4-6/ $\forall A.\ A \in DFA \iff A \in NFA$

$\Rightarrow$

$\Leftarrow$

DFA $\Rightarrow$ NFA/



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

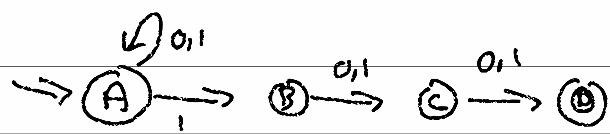
NFA $\delta': Q \times \Sigma_\varepsilon \to P(Q)$

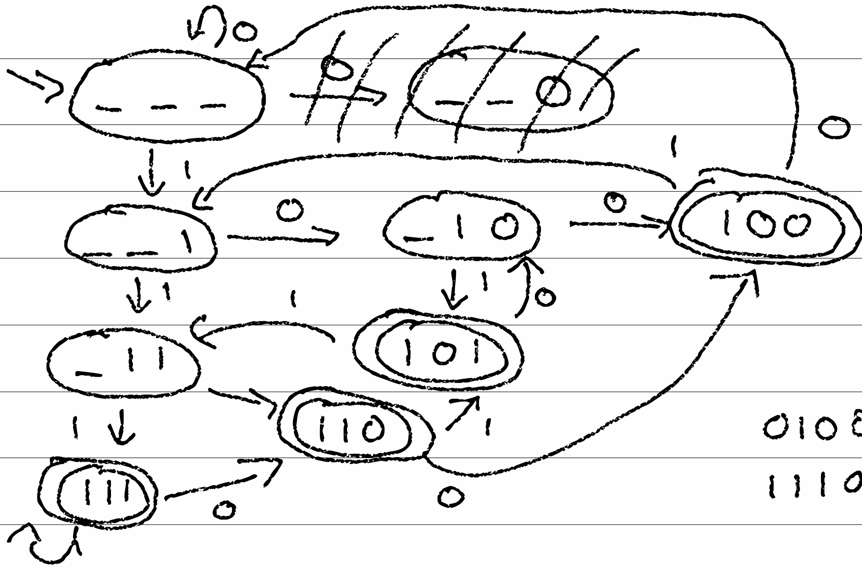
$$\delta'(q_i, \varepsilon) = \emptyset$$

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

NFA $\Rightarrow$ DFA/

3rd is 1





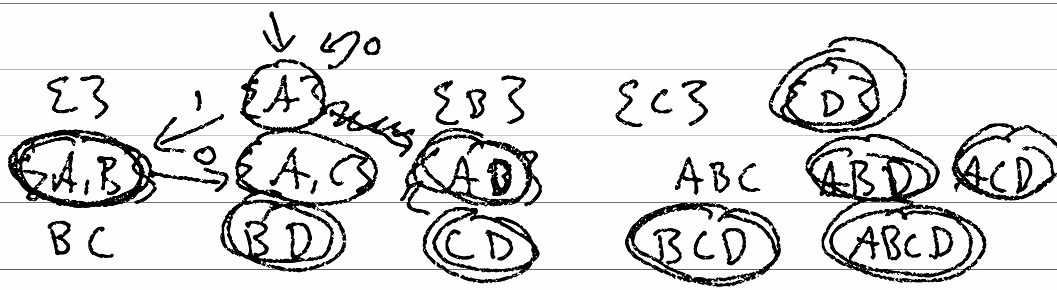
0100

111001

4–7/ in NFA $= (Q, \Sigma, q_0, \delta: Q \times \Sigma_\varepsilon \to P(Q), F)$

out DFA $= (Q', \Sigma, q_0', \delta': Q' \times \Sigma \to Q', F' \subseteq Q')$

$Q' = P(Q)$



$q_0' = \{q_0\}$

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

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

$\bigcup \delta(q_i, c)$

5-1/ $A \cup B$

$\delta_A : Q_A \times \Sigma \to Q_A$

$\delta_B : Q_B \times \Sigma \to Q_B$

$$\delta' : \underbrace{Q}_{(Q_A \times Q_B)} \times \Sigma \to Q$$

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

char

(p, c) => new Pair(deltaa(p.fst, c),
deltab(p.snd, c));

Pair<State, State>

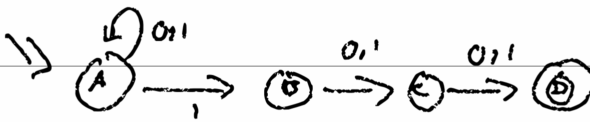
fst snd

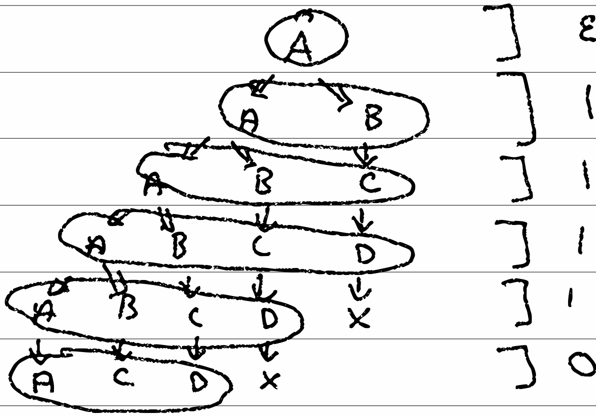
5-2 / NFA $\Rightarrow$ DFA

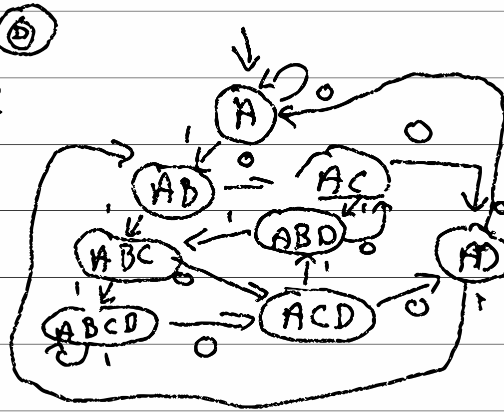
$(Q, \Sigma, q_0 \in Q,$
$\delta : Q \times \Sigma_\varepsilon \to P(Q),$
$F \subseteq Q)$

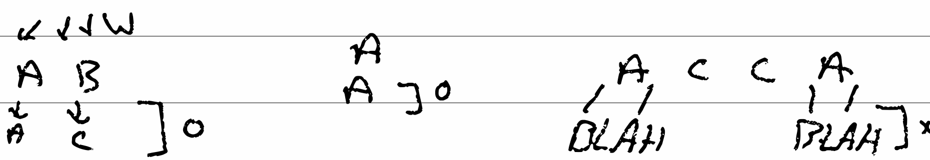
$\Rightarrow$

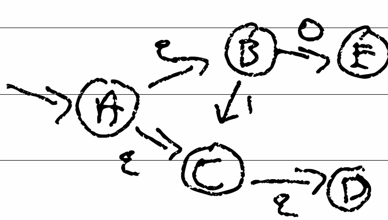
$(Q', \Sigma, q_0' \in Q'$
$\delta' : Q' \times \Sigma \to Q',$
$F' \subseteq Q')$











A B C D]

C D] 1

5-3

$E : Q' \to Q'$ — follow all
$P(Q) \to P(Q)$ ε-transitions

Trace Tree DFA

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

q_0' = the top of the tree
= the set that has only the first state
$= E(\{ q_0 \}) \in P(Q)$

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

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

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

= (set-of-qs →
foreach qi in set-of-qs
if dfa.F.apply(qi) then
return true
return false

5-4)

E (set <Q> qi)

queue <Q> ~~next~~ = ~~empty~~ qi

set <Q> seen = empty

while (nqt isn't empty)

~~add next.first to seen~~

δ(next.first , ε) add those

to next unless in seen

return seen

$E(A)$ = least fixed point of

$E^*(A)$

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