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}

{✓, ☺}

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

$\forall x, \quad x \in \{y\}$ 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"

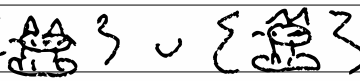
A set IS its membership

"1+1=2" $\in$ TS ↑ ?

"Is there a god?"

"Will Buffy be remade?"

All sets "constructed" via $\emptyset$, $\{x\}$, $\cup$ are finite.

$x \in$  $\cup$

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

$A \subseteq B$ iff $\forall x.\ x \in A \Rightarrow x \in B$ (↙ subset)

14) 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

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

chars chars

$\downarrow^0 \downarrow^1 \downarrow^2$

"0100001" = a string = s

length = 7 $s(0) = 0$ $s(1) = 1$ $s(2) = 0$

$\mathcal{U} = \Sigma^*$ ← special notation

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

epsilon = "" = the string w/ no characters

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

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

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

L5/ #1. Decide a data type to represent alphabets and characters.

```
Alphabet  = List<Character>
Character = Object  / void*
            we need equality
```

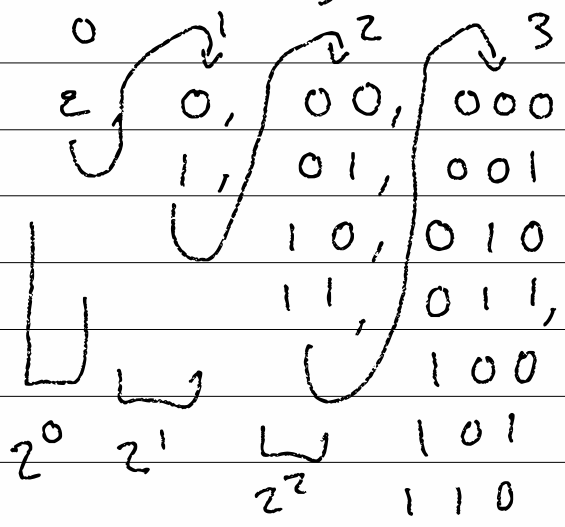
#2. Decide a data type for strings

```
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())));
```

$\{\Sigma\}$

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

$\Sigma = \{0,1\}$
" $= B$
new List<char>
{ Zero, One};



$|A|^i$ 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 = 11

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 $\{1, 2\}$

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

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

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

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

all even length strings $\subseteq P(\Sigma^*)$ $= \{\varepsilon, 00, 11, 01, \}$

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

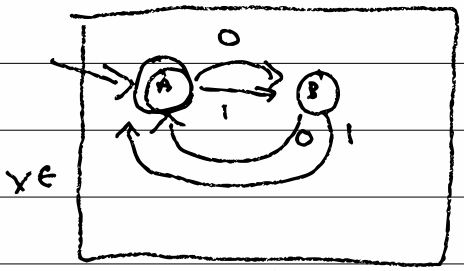
ALL	FIN
- True math	
- GIFs	
- Even strings	

$\emptyset \in FIN$

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

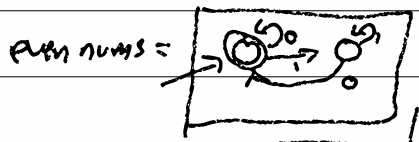
$A \in FIN \wedge B \in FIN \rightarrow A \cup B \in FIN$

All even strings =



ε or 11

Even nums =



DFA - a deterministic finite automata

○ - states $\{A, B\}$

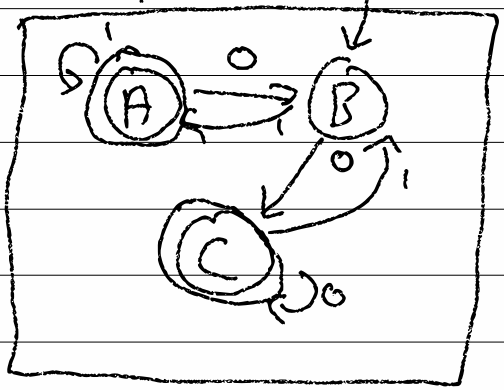
→○ - start state A

◎ - accepting states $\{A\}$

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

x - labels are Σ

	0	1
A	B	B
B	A	A



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

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

	0	1
A	B	A
B	C	A
C	C	B

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

DFA configuration = $Q \times \Sigma^*$

$[\hat{q}]\, w$

config update function : config × DFA ⇒ config

$[q]w \rightarrow [q']w'$

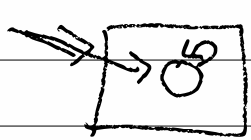
$[q_i]\, x\, y \Rightarrow [q_j]\, y$ iff $\delta(q_i, x) = q_j$

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

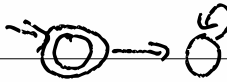
$0110 \in EvenLen$ iff $[A]0110 \Rightarrow [B]110 \Rightarrow [A]10 \Rightarrow [B]0 \Rightarrow [A]\varepsilon$ $A \in \{A\}$ ✓

```
class DFA ε
  ... Q, ε, 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); } }
```

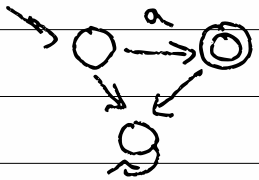
2-4/



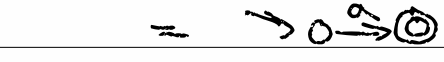
$= \emptyset =$

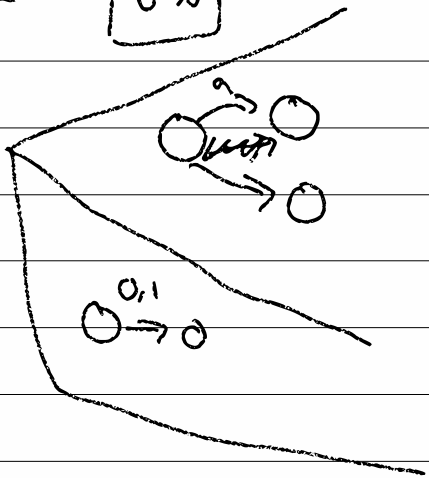


$= \{\varepsilon\}$

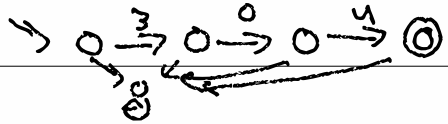


$= \{a\}$



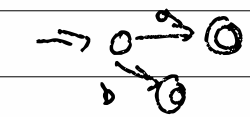


$\{304\} =$

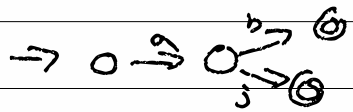


$A \cup B \in DFA$ (i.e. $A \in DFA$ and $B \in DFA$)

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

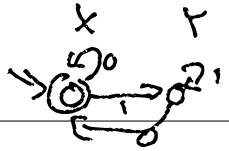


$\{a, ab\}$

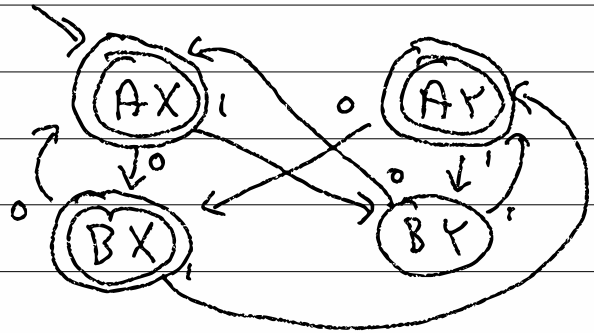


2-5] Even Len

Is Even



∪



ε ✓

00 ✓

11 ✓

0 ✓

110 ✓

∩

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

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

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

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

$X = A \cup B$

$Q_X = Q_A \times Q_B$

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

$F_X = F_A \times F_B$ — ∩

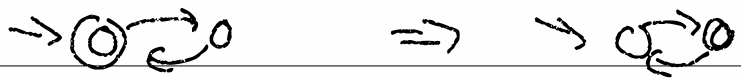
$F_A \times Q_B \cup Q_A \times F_B$ — ∪

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

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

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

Even Len odd len



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

or F^c (wrt Q)

complement

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

3-1) DFA => example or false

DFA :

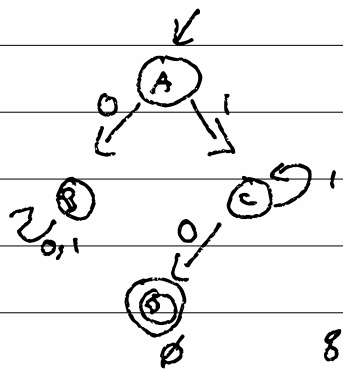
Q : State -> Bool

Σ : list of characters

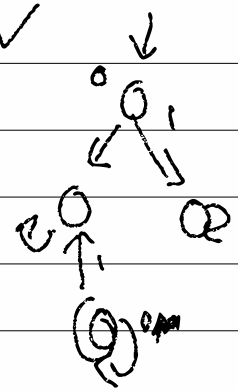
q_0 : state

δ : (state x (char) -> State

F : State -> Bool



10 ✓



Q	Ø	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, 'is possible.'
{D}	{A, C, D}	[D]	D => C, 0
{}	{A, B, C, D}	[]	or if not, 'No!'

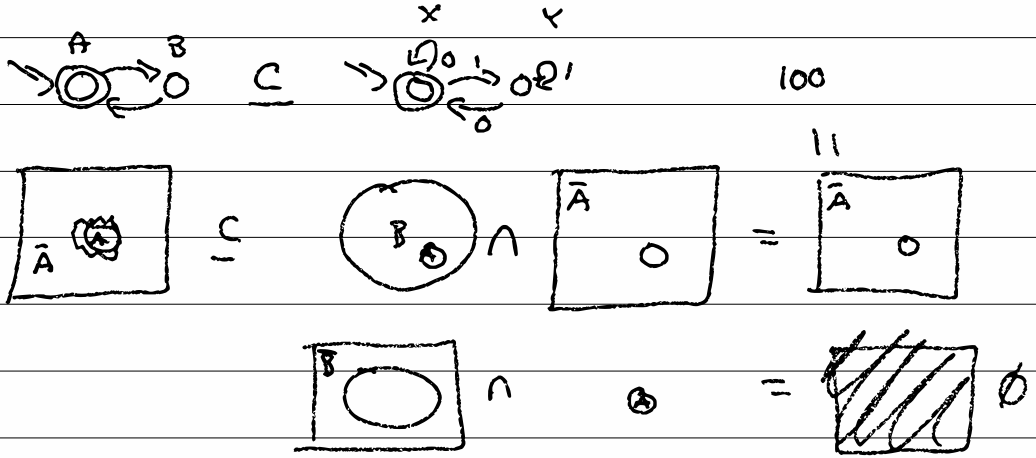
10

3-2/ subset

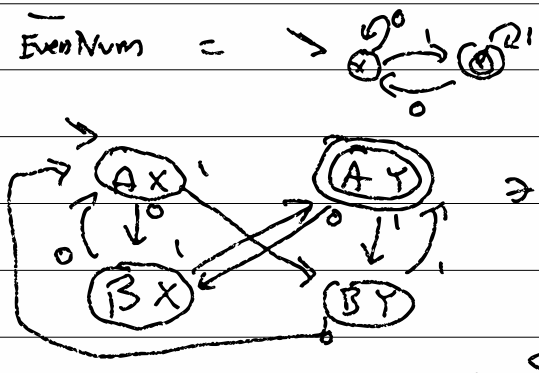
$A \subseteq B$ iff $\forall x \in U.\ x \in A \Rightarrow x \in B$ ↗ $\exists x \in U.\ x \in A \wedge x \notin B$

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

↳ finite means naive works!



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



⇒ 01

$\not\equiv \phi$

model checking ↗ Soundness : model $\subseteq$ theory

completeness : theory $\subseteq$ model

model = theory

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

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

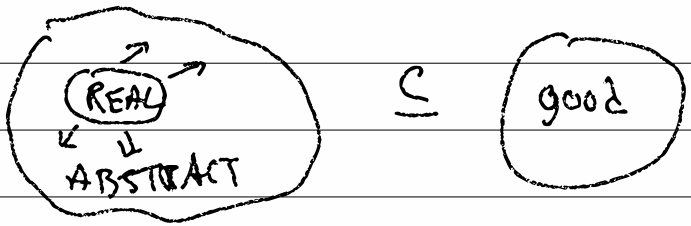
if $x > 0$ then

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

o.w

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

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



3-4) Finite = $\emptyset$ $\{x\}$ $A \cup B$ A^c $A \cap B$ $A \circ B$ $\in$ DFA

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

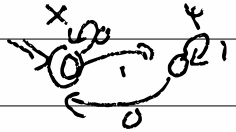
$abcd = ab \circ cd$

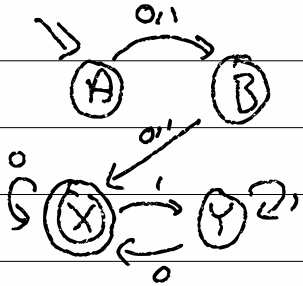
$\{j\} \circ \{m, n\} = \{jm, jn\}$

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

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

3-5/ Even Len ∘ Even Num



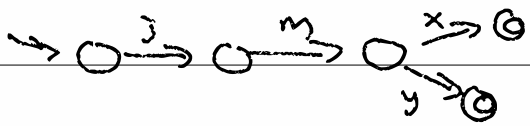


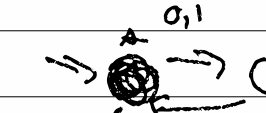
00110 ✓

0011 ×

~~0011~~

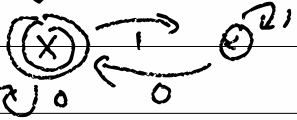
{ j m } ∘ { x, y }





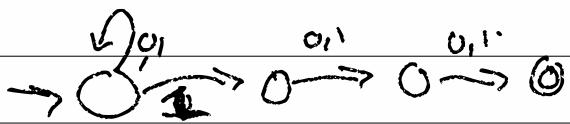
Σ = {0, 1}

maybe



x ∈ 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

dd world: the next step was obvious
$\delta: Q \times \Sigma \Rightarrow Q$

new world: crazy options
- do you even read actions?
- which path do you take?

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

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

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

epsilon

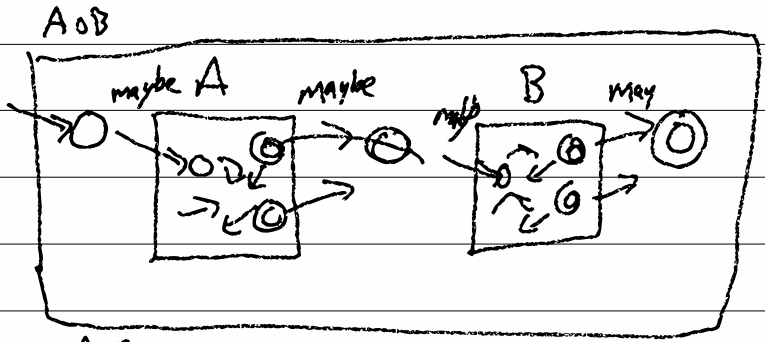
$\Sigma \quad \Sigma \quad e \quad (\varepsilon)$

4-1/ $A \circ B \in DFA$ iff $A \in DFA$

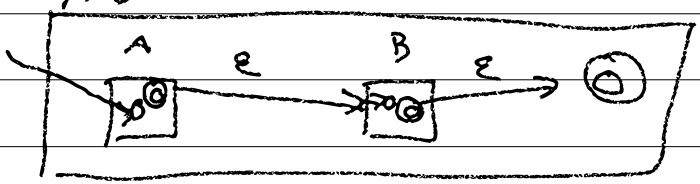
A^* $B \in DFA$

NFA (N - non D - deterministic)

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



maybe written as "ε"



What does (NFA) this mean?

NFA ⟷ DFA

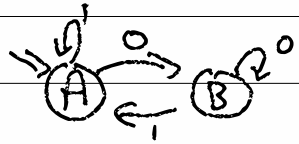
4-2 What do NFAs mean?

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

$\cup \rightarrow$ Bool

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

config $= \Sigma^* \times Q$



$\Sigma^* \Rightarrow Q^*$

0110 $\Rightarrow$ ABAAB $\rightarrow$ a trace

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

01~~1~~0 $\Rightarrow$ (0,B) (1,A) (1,A) (0,B)

Σ_ϵ ~~may~~ $= \Sigma \cup \{\epsilon\}$

0A 1A 1A 0B $\Rightarrow$ N

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

valid q_i ε = Y

valid q_i (c, q_j) : more = if $\delta(q_i, c) == q_j$ valid q_j more

Nvalid? : $Q \times (\Sigma_\epsilon \times Q)^* \Rightarrow B$

Nvalid q_i ε = Y

O.W. N

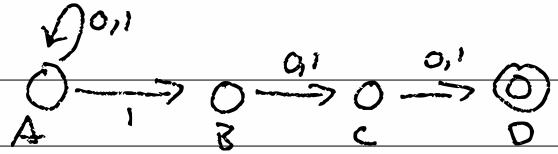
Nvalid q_i (c, q_j) : more =

if q_i ϵ $\delta(q_i, c)$ then — Nvalid q_j more

O.W. N

Ag 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 $\varepsilon = \varepsilon$
(0,B)(1,C)(1,D)(0,D)	X	str $(c, -)$: move c ∘ str move

$\delta(A,0) = \{A\}$ $\quad \delta(D,0) = \emptyset$

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

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

valid q_0 t $= Y$

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

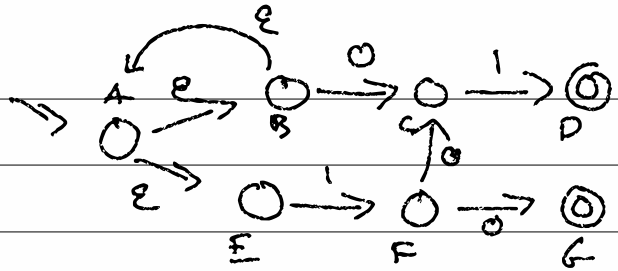
NFA-accepts : $\Sigma^* \to Y/N$

figure all possible traces

check if valid and is strings match

check if last is in $\in F$

4-4



(ε, B) (0, C) (1, D) 01 = ε0001

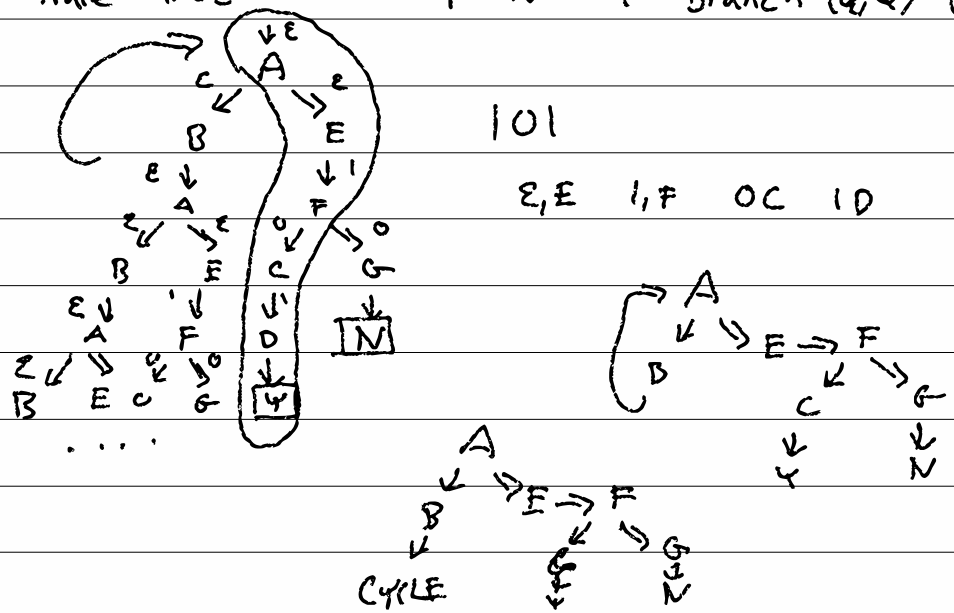
(ε, E) (1, F) (0, C) (1, D) 101 = ε010001

(ε, E) (1, F) (0, G) 10 = ε0100

(ε, B) (ε, A) x where x is valid

⇒ valid

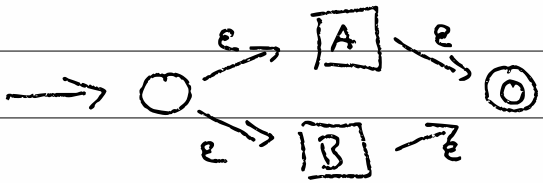
Trace Tree = Y | N | Branch (ε, Q) (List TT)



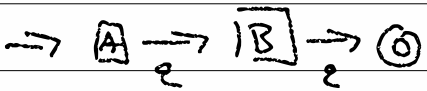
Forking model of NFAs (make TT)

Back-tracking model (explores TT)

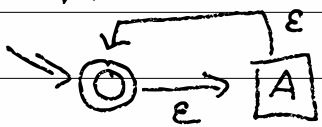
4-5) $A \cup B$



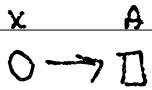
$A \circ B$



A^*



X A



State X transitions to THE start state of A

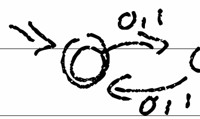
All accepting states of A transition to X

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

$\Rightarrow$

$\Leftarrow$

DFA ⇒ NFA/



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

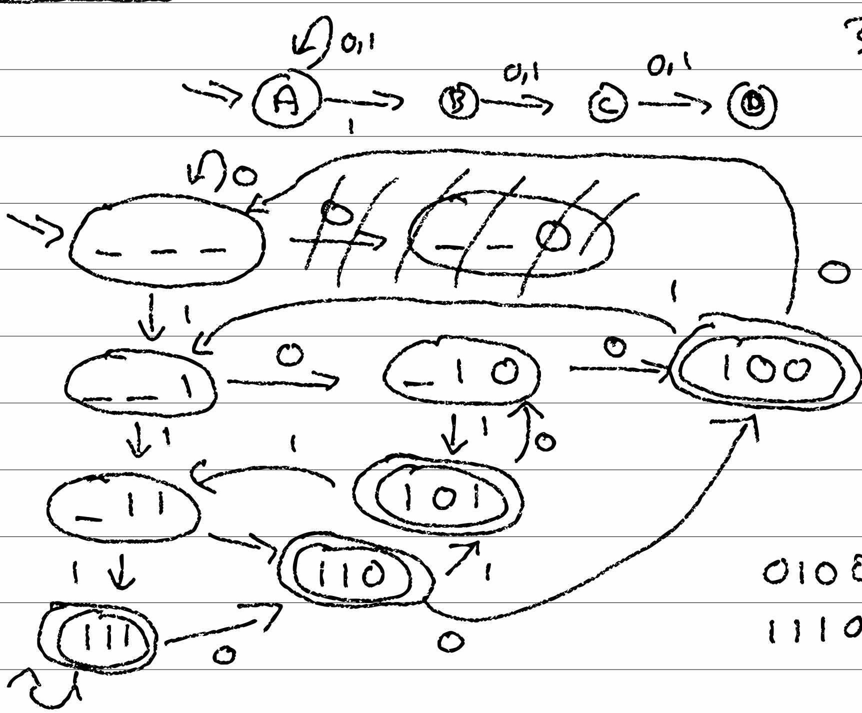
NFA $\delta': Q \times \Sigma_\varepsilon \to P(Q)$

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

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

NFA ⇒ DFA/

3rd is 1



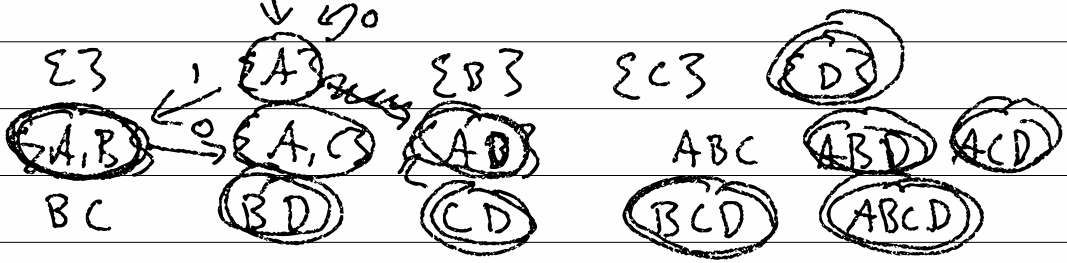
0100

111001

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

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

$Q' = P(Q)$



$q_0' = \{q_0\}$

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

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

5-1 / $A \cup B$ $\quad \delta_A : Q_A \times \Sigma \Rightarrow Q_A$

$\delta_B : Q_B \times \Sigma \Rightarrow Q_B$

$\delta' : Q \times \Sigma \Rightarrow Q$

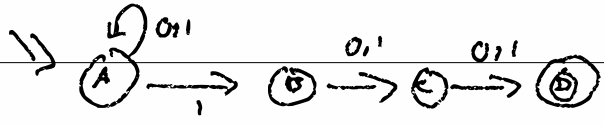
$Q = (Q_A \times Q_B)$

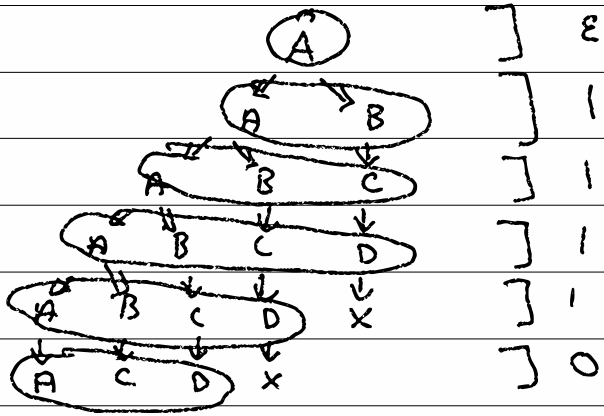
$\delta((q_a, q_b), c) = (\delta_A(q_a, c), \delta_B(q_b, c))$

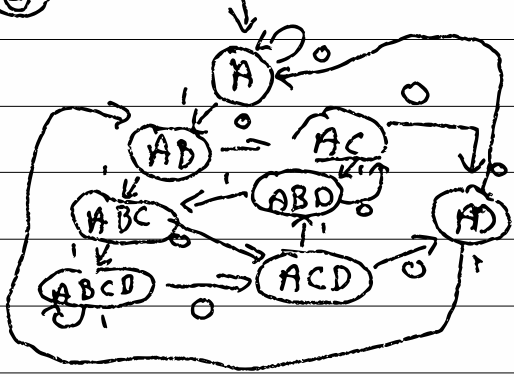
```
                    Char
(P, C) => new Pair( deltaa(p.fst, c),
Pair<State, State)    deltab(p.snd, c));
  fst
  snd
```

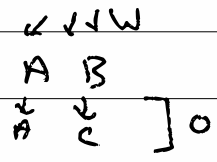
5-2/ NFA ⇒ DFA

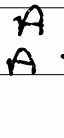
$(Q, \Sigma, q_0 \in Q, \delta: Q \times \Sigma_\varepsilon \to P(Q), F \subseteq Q)$ ⇒ $(Q', \Sigma, q_0' \in Q', \delta': Q' \times \Sigma \to Q', F' \subseteq Q')$

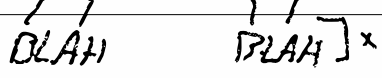


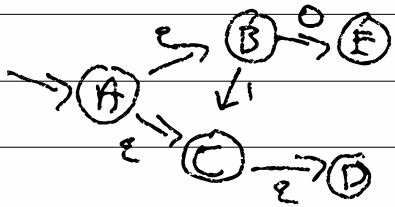












5-3/

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

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 tree 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 q_i in set-of-qs
if dfa.F.apply(q_i) then
return true
return false

5-4/

```
E (set<Q> qi)
    queue<Q> next = qi
    set<Q>   seen = empty
    while (next 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 $E^*(A)$

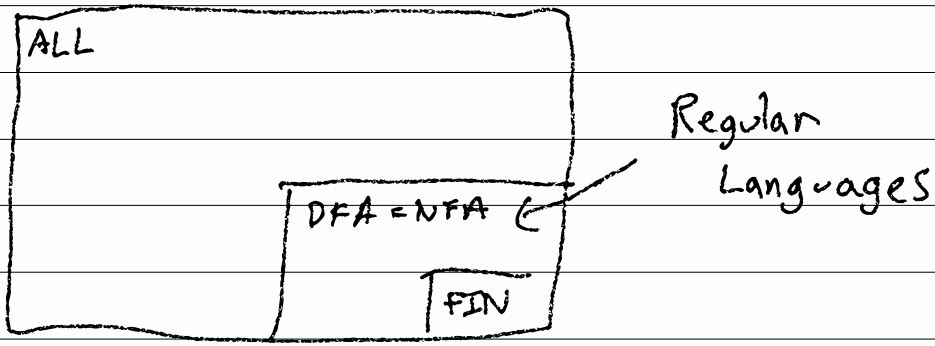
$E^*(A) = A \cup \bigcup_{q_i \in A} \delta(q_i, \varepsilon)$

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

$N = D$

$\forall D \in DFA.\ \exists N \in NFA.$

$D = N$

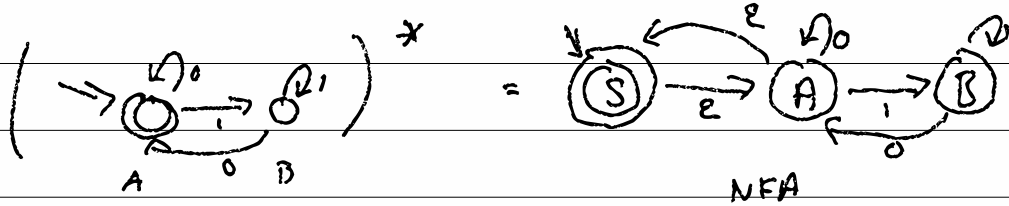


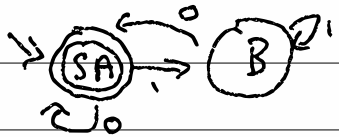
program f ....

program g ....

" f ; g "

compositional



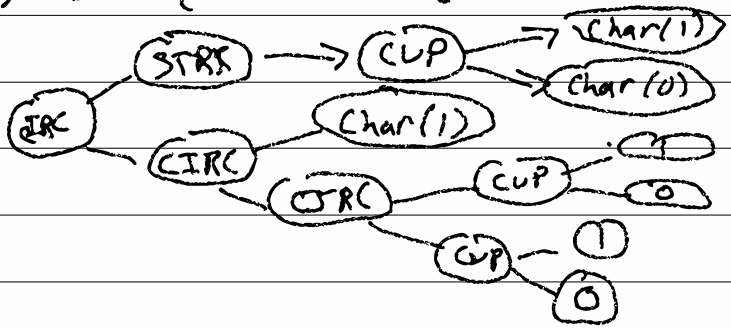


6-2 / Regular Expressions

re :=		
	ε	EMPTY
	$\emptyset$	NULL
	c	Char
	re ∪ re	CUP
	re*	STAR
	re ∘ re	CIRC

```
interface RegEx { }
class RE_Empty() impl REgex {...
    RE_NULL()
    Re_Char (Char c)
    Re_cup (Regex lhs, Regex rhs)
```

$(1 \cup 0)^* \circ 1 \circ (1 \cup 0) \circ (1 \cup 0)$ = "3rd from end is 1"




```
new Circ( new Star(
    new Cup(
        new Char(1),
        new Char(0))
    ...
)
```

6-3| $L : RE \Rightarrow$ ~~#~~$LL = P(\Sigma^*)$

file the language it describes (or some)

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

$L(\emptyset) = \emptyset$

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

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

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

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

```
class Cup { L() {
        return union(lhs.L(), rhs.L()) }}
```

compile : RE → NFA

compile (ε) = ⇒◎

compile (∅) = ⇒○

compile (c) = ⇒○ —c→ ◎

compile (r ∪ r') = ⇒ ○ —ε→ [r] —ε→ ◎ ; ○ —ε→ [r'] —ε→ ◎

compile (r ∘ r') = ⇒ [r] —ε→ [r'] —ε→ ◎

compile (r*) = ⇒ ○ —ε→ [r] —ε→ ○ ; ○ —ε→ ◎

6-4)

" " $= \varepsilon$

——— $= \emptyset$

" c " $= c$

" xyz* " $= x \circ y \circ (z^*)$

" (xyz)* " $= (x \circ y \circ z)^*$

" xyz " $= x \circ y \circ z$

" [abc] " $= (a \cup b \cup c)$ $(a, b, c \in \Sigma)$

" (a | b | c) " $= \Rightarrow$ $(a, b, c, \in \Sigma^*)$

[0|2]

(zero | one | ~~boo~~)

• $= \Sigma$

" . * \. mjs " $= \Sigma^* \circ '.' \circ 'm' \circ 'j' \circ 's'$

6-5/

gen : RE → Σ* or false

gen ε = ε

gen ∅ = FALSE

gen c = 'c'

flip coin

gen x ∪ y = gen x H gen y

gen x ∘ y = gen x ∘ gen y

gen x* = ε

= gen (ε ∪ x ∘ x*)

"ajs"

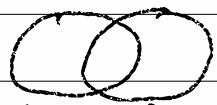
equal : RE × RE → Bool

equal x y =

NFA2DFA(compile x)

NFA2DFA(compile y)

dfa equality?



$(\bar{A} \cap B) \cup (A \cap \bar{B}) = \emptyset$

6-6/

$x + 0 = x$ $\qquad$ $x = x$

$x \cdot 1 = x$

$\dfrac{a = b}{a + x = b + x}$ $\qquad$ $\dfrac{a = b \quad x \neq 0}{ax = bx}$

$2(3x + 17) = 6x + 34$ $\qquad$ "algebra"

$3x + 17 = 3x + 17$

$3x = 3x$

$x = x$ $\qquad$ $1776 = 1776$?

~~1776~~

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

$\emptyset \circ x = x \circ \emptyset = \emptyset$

$\varepsilon \circ x = x \circ \varepsilon = x$

$\emptyset^* = \varepsilon = \varepsilon \cup \emptyset \circ \emptyset^*$ $\qquad$ $x^* = \varepsilon \cup x \circ x^*$

$= \varepsilon \cup \emptyset = \varepsilon$

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

NFA 2 DFA $\qquad$ in: N states $\quad (Q)$

out: 2^N states $\quad (P(Q))$

6-7/ size : RE → Nat

size ∅ = 1

size ε = 1

size c = 1

size (x ∪ y) = sz x + sz y + 2

size (x ∘ y) = sz x + sz y + 1

size (x*) = sz x + 1

x ∘ (y ∪ z) = x ∘ y ∪ x ∘ z

↑

---

emacs *.c

↓ bash

emoaocos ε*∘a∘c

↓ ↓ compile

→0→0→0→ →0→0→0→0

↓ ↓ NFD

↓ 304.c main.c

↓

emacs

dir

304.c

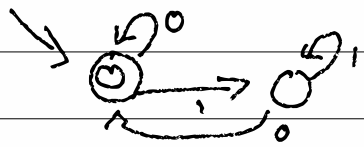
awesome-quotes.txt

mycat.jpg

main.c

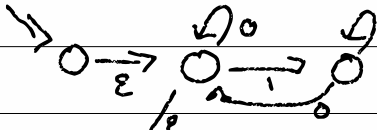
+

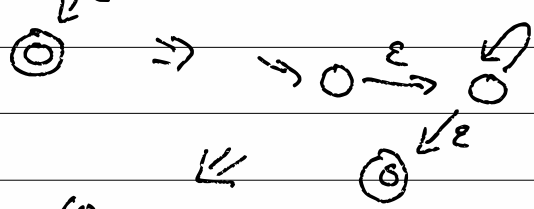
6-8/ décompile : (NFA) ou DFA ⇒ RE

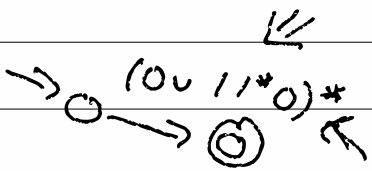


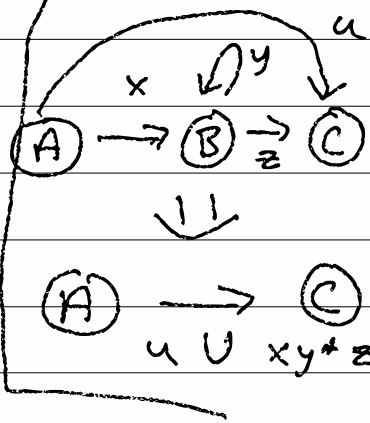
$\Sigma^* \circ 0 \cup \varepsilon$

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









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

ε 0 10 111110

01110111100

1110001101101010

G-9 / decompile : N-state NFA

⇒ RE

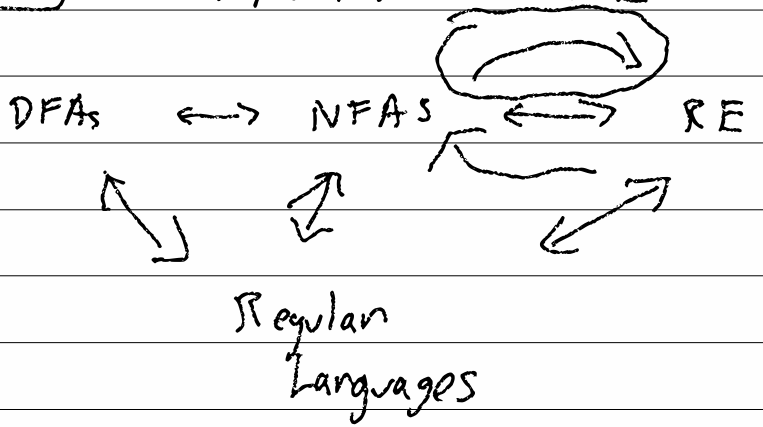
START : N-NFA ⇒ (N+2)-GNFA

RIP : (N+1)-GNFA ⇒ N-GNFA

END : 2-GNFA ⇒ RE

decompile m = end ∘ rip^n ∘ start (n)

7-1 DFAs/NFAs → RE



NFA → RE

IN : n-NFA → (n+2)-GNFA

RIP^n : (n+1)-GNFA → n-GNFA

OUT : 2-GNFA → RE

GNFA = $(Q, \Sigma, q_s, q_e, \Delta : (Q - q_e) \times (Q - q_s) \to RE(\Sigma))$

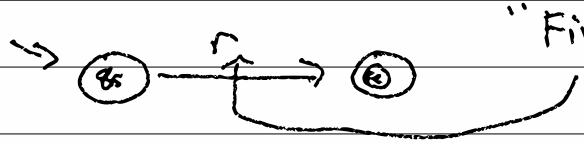
($q_s \in Q$, $q_e \in Q$)

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

OUT : 2-GNFA → RE

$(\{q_s, q_e\}, \Sigma, q_s, q_e, \{((q_s, q_e), r)\})$

$= r = \Delta(q_s, q_e)$

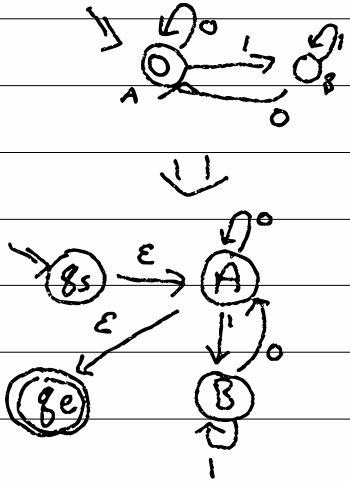


7-2) IN: NFA ⇒ GNFA

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

$(Q', \Sigma, q_s, q_e, \Delta: (Q' - q_e) \times (Q' - q_s) \to RE)$

$Q' = Q \cup \{q_e, q_s\}$



$\Delta(q_i, q_j) = r$

$\Delta(q_s, q_0) = \varepsilon$

$\Delta(q_s, q_j \neq q_0) = \emptyset$

$\Delta(q_f \in F, q_e) = \varepsilon$

$\Delta(q_i \notin F, q_e) = \emptyset$

$\Delta(q_i, q_j) = \bigcup \{ c_{i,j} \mid \delta(q_i, c) \ni q_j \}$

7-3/ RIP = (n+1)-GNFA → n-GNFA

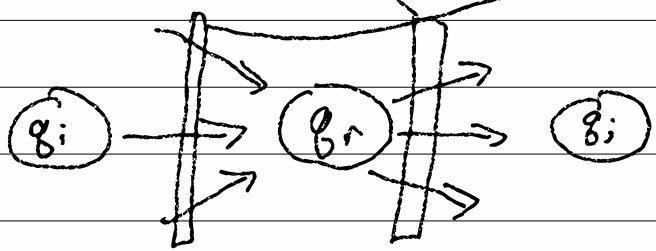
main → (Q, Σ, qs, qe, Δ) → (Q', Σ, qs, qe, Δ')

jay main → Q = Q' ∪ { q gonna be killed }

r → g → exit jay → exit

Δ' : $(Q' - q_e)$ × $(Q' - q_s)$ → RE ($q_r \notin$ = $q_r \notin$)

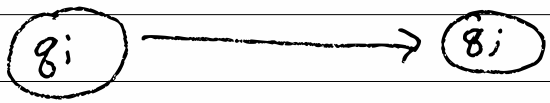
Δ : $(Q - q_e)$ × $(Q - q_s)$ → RE ($q_r \in$, $q_r \in$)



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

~~$x \circ z$~~

$\Delta'(q_i, q_j)$ =

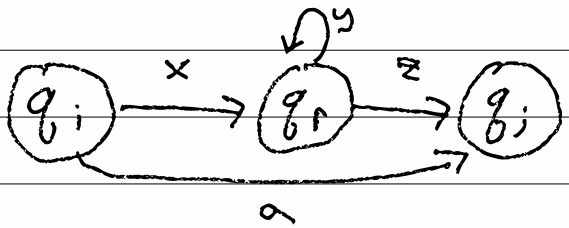


$x = \Delta(q_i, q_r)$

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

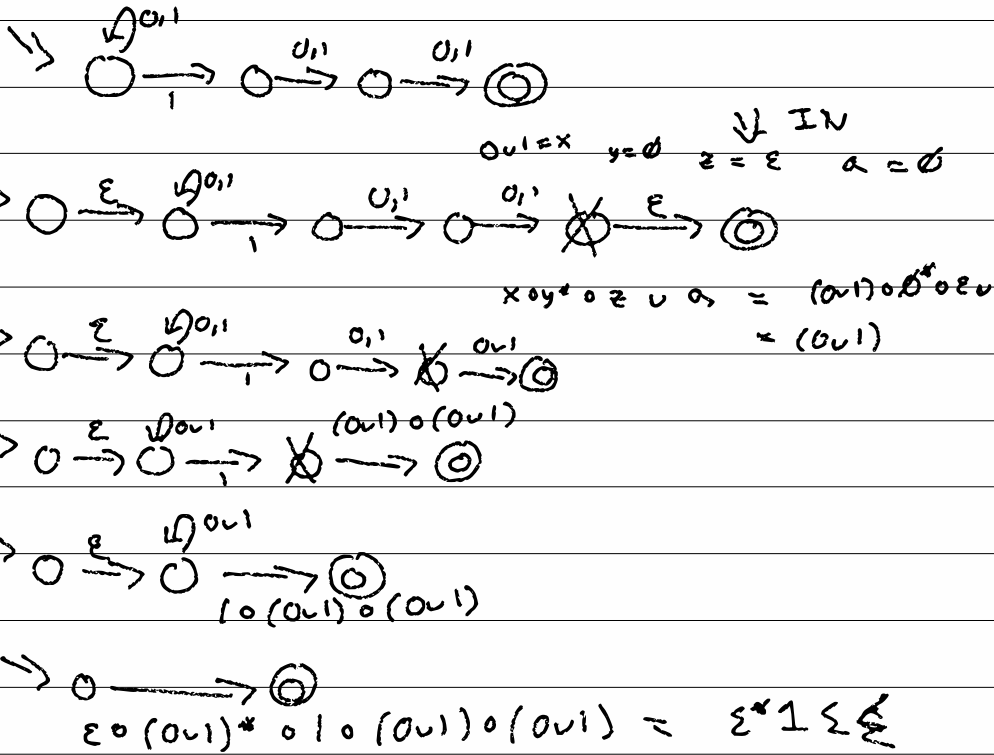
$z = \Delta(q_r, q_j)$

$a = \Delta(q_i, q_j)$

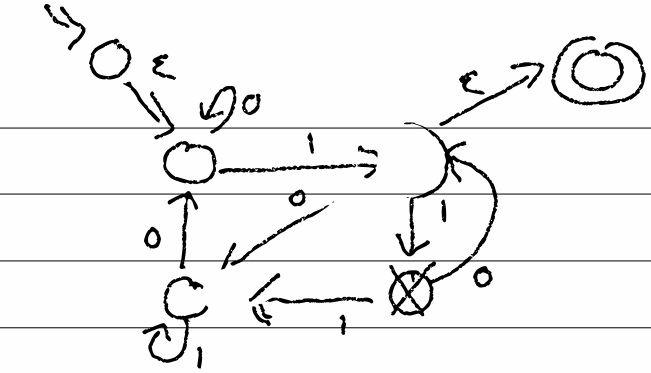


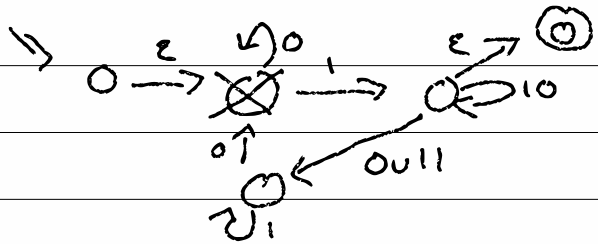
7-4/

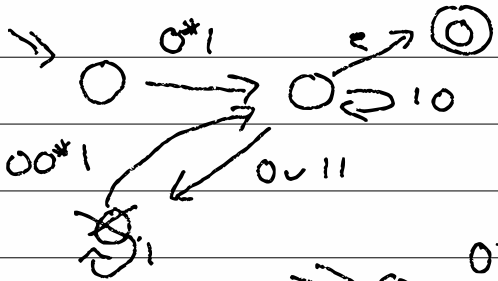
```
int main() {
    int x = 8;
    int y = f(x, 2);
    ....
    int y = x + 2 * 2;
}

int f(int z) {
    int g;
    return z + 2*g;
}
```

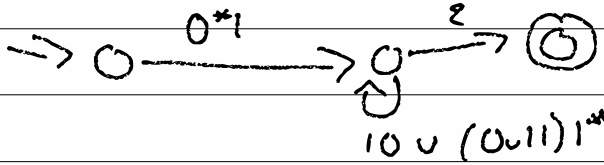


7-5/









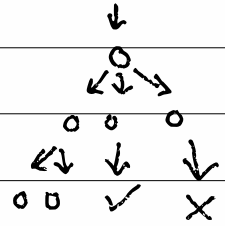
$(0^*1)(10 \cup (0 \cup 11)1^*00^*1)^*$

8-1 / Char (char c)

Object — Epsilon ()

UTF-8

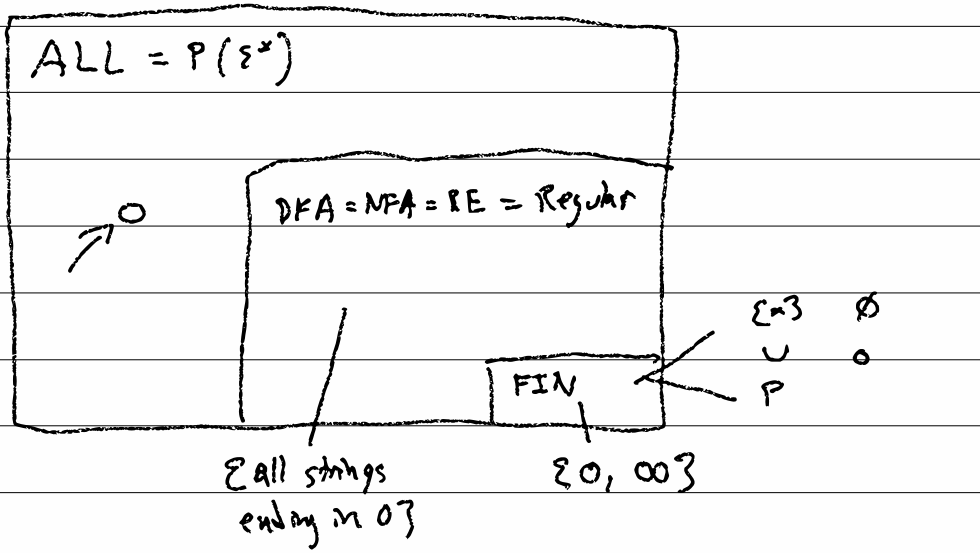
TT



TT = Succ | Fail

| Branch State List (TT)

Char × TT



$A \in REG \iff \exists d \in DFA.\ L(d) = A$

$\neg\, A \in REG \iff \neg(\ \nearrow\ ) \iff \forall d.\ d \in DFA \wedge L(d) \neq A$

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

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

8-2/ How can I know stuff about infinite sets?

$\forall x \in A,\ P(x)$ $\quad P : DFA \to Prop$

$\Rightarrow$ $\quad q_0 \in Q$

$P : \mathbb{P}(\Sigma^*) \to Prop$

$\neg P(B)$ (where $B \in \mathbb{P}(\Sigma^*)$ and we "hope" B isn't in DFA)

0. What is P?
1. Prove $\forall A \in REG.\ P(A)$ ← Pumping lemma
2. Prove $\exists B \in ALL.\ \neg P(B)$

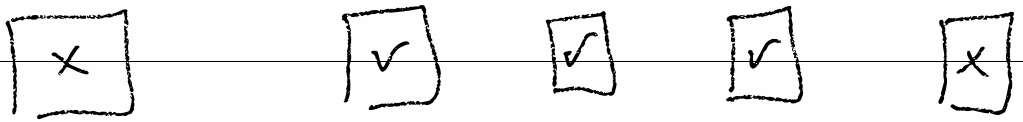
---

conclude $B \notin REG$.

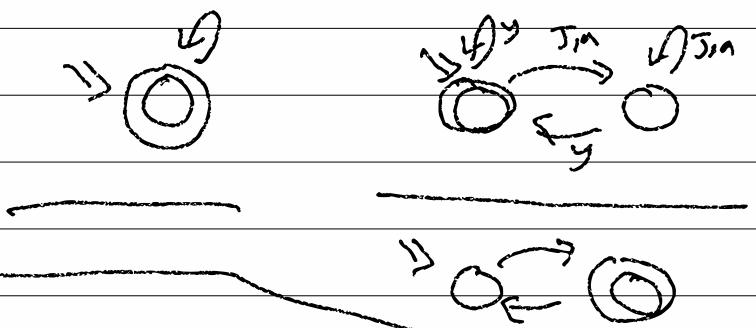
$\Sigma = \{\ldots\}$

8-3/

"Jay" $\in$ d $\qquad \Sigma \ni \{J, a, y\}$



$|d.Q| = ?$



Daphne wins

pick a number of states ; 4

→ she picks a char

I say what state we go to

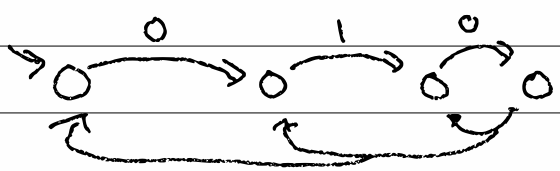
I win if I never say same state

she wins if I repeat

How many turns to win?

4 ⇒ N ⇒

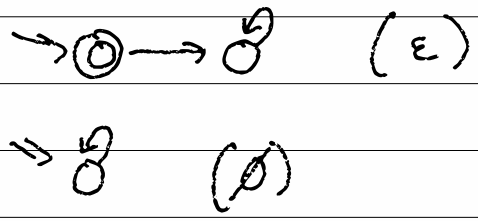
$|Q|$

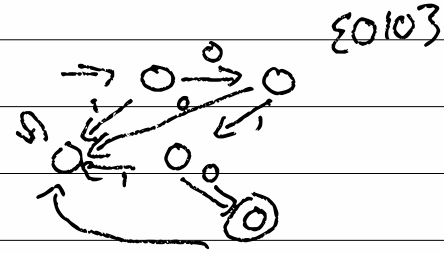


Ex-4/ $\delta : Q \times \Sigma \to Q$ Total Fun

from $0,1$

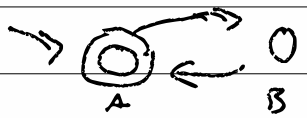
All DFAs have a loop





Some have "exciting" loops

$\exists$ s ∈ L(DFA) . s = ...

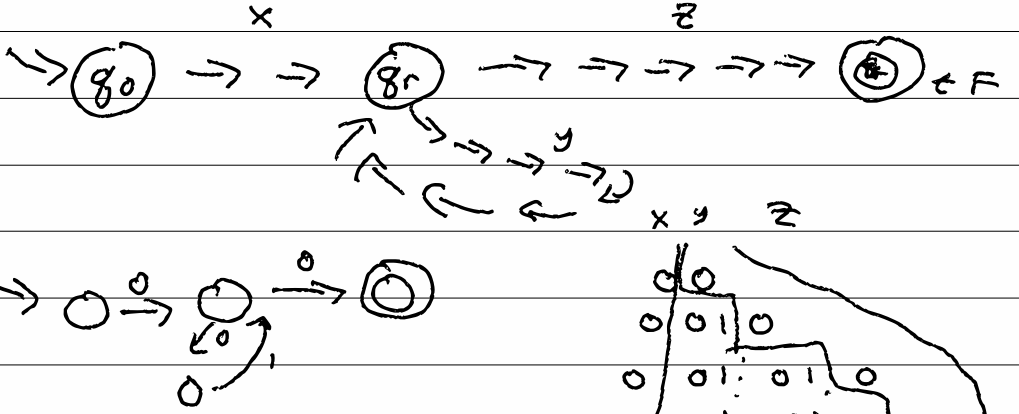


(Even lengthed)

ε = A

= ε ∘ ε ∘ ε

0110 = ABABA = ε ∘ 01 ∘ 10



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

They are all finite

$L(m) \in FIN$

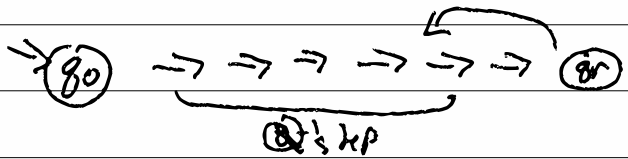
If it does have one ...

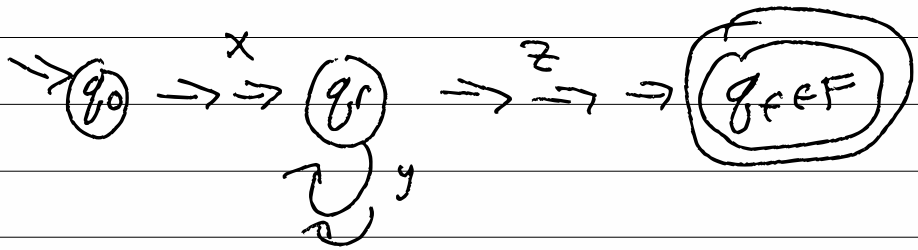
then it must be infinite (regular)

If there is an exciting loop ...

what is the shortest string to "find" it in?

$|s| \leq |Q|$ $(s \in L(\delta))$





$\in$ ALL (a language) $(\mathbb{P}(\Sigma^*))$

8-6/ RPP(A) = Regular Pumping Property

$$\exists p \in \mathbb{N}. \quad // \; p = |Q|$$
$$\forall (s \in A \mid |s| \geq p)$$
$$\exists (x, y, z \in \Sigma^* \mid s = xyz \wedge |xy| \leq p \wedge |y| > 0)$$
$$\forall i \in \mathbb{N}.$$
$$x \circ y^i \circ z \in A$$

$\forall d \in DFA. \exists r \in RE. L(d) = L(r)$ — Coq

$\neg RPP(B) :=$

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

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

$\forall (x, y, z \in \Sigma^* \mid s = xyz \wedge |xy| \leq p \wedge |y| > 0 \Rightarrow$

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

$x \circ y^i \circ z \;\&\; B$

8-7] Need: an infinite space problem

0^*1

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

$= 2^{32}$ uint2_t

$\forall n \in N$,

$0^n 1^n \in B$

```
uint count = 0;
while (in == 0) count++
while (in == 1) count--
if (in == EOF)
return count == 0 ;
{ ret false } }
```