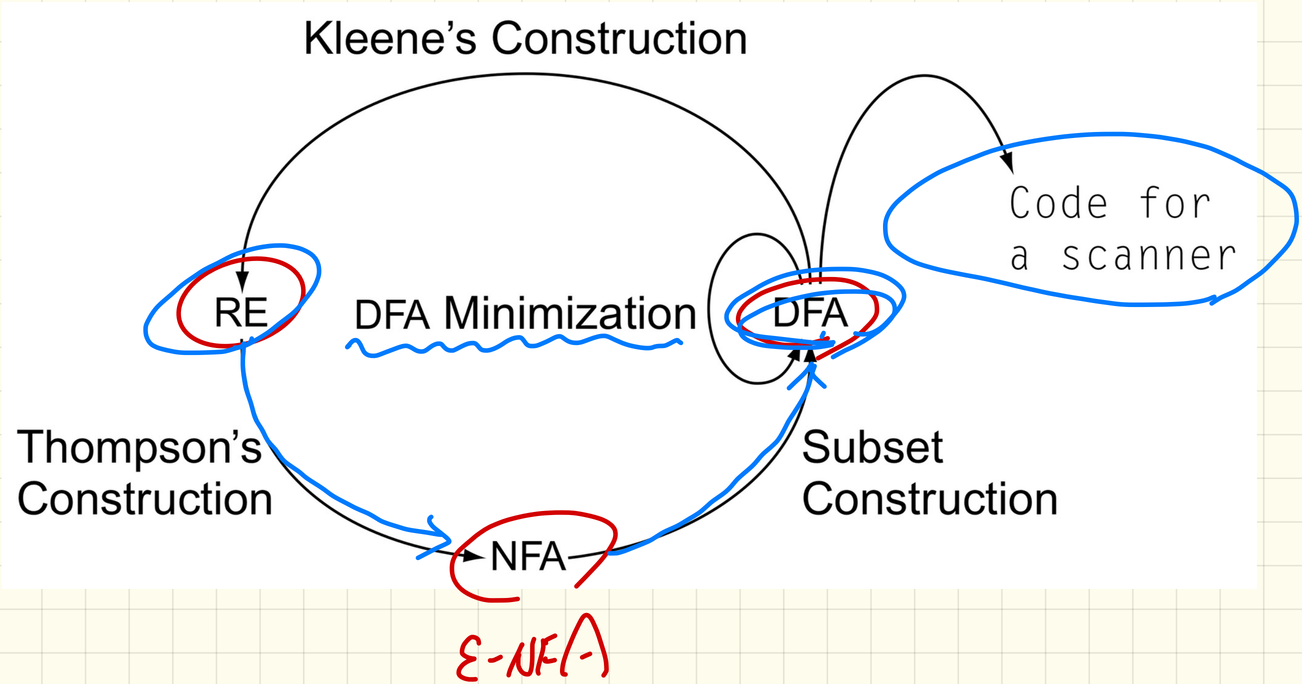
Lecture 3

Monday January 13

Scanner: Formulation & Implementation



Set Comprehension

$$\{ \underbrace{2x}_{\text{term}} \mid \underbrace{1 \leq x \leq 10}_{\text{condition}} \}$$

$$= \{2, 4, 6, \ldots, 20\}$$

$$\Sigma_{bin}^{*} = \Sigma_{bin}^{0} \cup \Sigma_{bin}^{1} \cup \cdots$$

$$\Sigma_{bin}^{0} = \{\varepsilon\}$$

card. 1

$\varepsilon \approx$ ""

× $0|0|0 \in \Sigma_{bin}$

$\underline{00} \notin \Sigma_{bin}^{1}$

✓ $0|0|0 \in \Sigma_{bin}^{*}$

& strings of all possible lengths

$\{0, 1\}$

$\underline{00} \in \Sigma_{bin}$

$$\Sigma_{bin} = \{\underline{0}, \underline{1}\}$$

$1, 0$

$$\Sigma_{bin}^{1} = \{0, 1\}$$

$$\Sigma_{bin}^{2} = \{00, 01, 10, 11\}$$

alphabet

$$\Sigma_{my} = \{00, 11\}$$

single single

$$0011 \in \Sigma_{my} \quad \times$$

$$00 \in \Sigma_{my} \quad \checkmark$$

$$\Sigma_{my}^{2} = \{0000, 0011, 1100, 1111\}$$

$$\underbrace{\{a \ldots z\}^5}_{26}$$

$$\{a \ldots z\}^0$$

$$\{\varepsilon\}$$

$$26 \quad 26 \quad \ldots \quad 26$$

$$26^5.$$

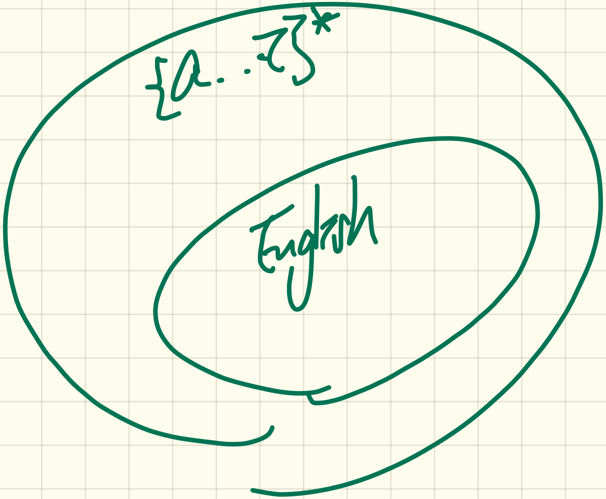
$\underline{\{a..z\}^*} = \text{English} \;?$

$aa \in$

$aa \notin$

$w \leftarrow L.$

$RL.$



input program

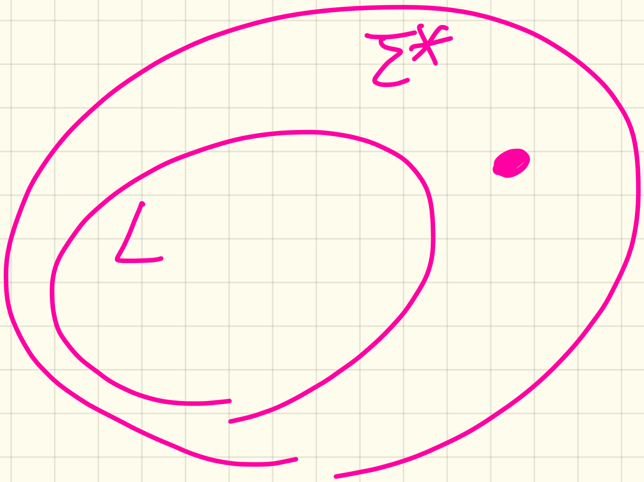
$\overset{w}{\boxed{\text{class}}}$ A {

⋮

}

$\forall : L \subseteq \Sigma^*$

$$\underline{w \in L} \quad \underset{\not\Longleftarrow}{\Longrightarrow} \quad \underline{w \in \Sigma^*}$$



$+$

$E \mid F$

$0 \mid 1^*$ $\qquad$ $(01)^*$

A $\qquad$ B

011 $\qquad$ 0101

0 $\qquad$ ε

Regular Language Operations

Cardinalities?

1. Union

$$|L \cup M| = \{w \mid w \in L \vee w \in M\}$$

LM

$L = \{a, b\}$

$M = \{1, 2, 3\}$

2. Concatenation

$$LM = \{xy \mid x \in L \wedge y \in M\}$$

$|LM| = 6$

$\{a1, a2, a3, b1, b2, b3\}$

3. Kleene Closure (or Kleene Star)

$$L^* = \bigcup_{i \geq 0} L^i$$

$L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$

$= L^*$

Constructions of REs

Base Case:

- Constants ϵ and $\varnothing$ are regular expressions.

$$\begin{aligned} L(\epsilon) &= \underline{\{\epsilon\}} \\ L(\varnothing) &= \varnothing \end{aligned}$$

- An input symbol $a \in \Sigma$ is a regular expression.

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

RE op.

Recursive Case Given that E and F are regular expressions:

- The union $E + F$ is a regular expression.

$$L(\boxed{E} + \boxed{F}) = L(E) \cup L(F)$$

E | F

- The concatenation EF is a regular expression.

$$L(EF) = \underline{L(E)L(F)} = \{xy \mid x \in L(E) \wedge y \in L(F)\}$$

- Kleene closure of E is a regular expression.

RE operator

$$L(E^*) = (L(E))^*$$

RL operator

- A parenthesized E is a regular expression.

$$L((E)) = L(E)$$

$$\underline{\emptyset^{*}} = \boxed{\emptyset^{0}} \cup \underline{\emptyset^{1}} \cup \underline{\emptyset^{2}} \cdots$$

regular expression

$\|$

$L(\emptyset) = \emptyset$

$$= \underline{\{\varepsilon\}} \cup \underbrace{\{x \mid x \in \emptyset\}}_{\emptyset}$$

$$\cup \underbrace{\{xy \mid x \in \emptyset \wedge y \in \emptyset\}}_{\emptyset}$$

$\underline{\{\boxed{\emptyset}\}}$ vs. $\underline{\emptyset}$

$$= \{\varepsilon\} .$$

RE: Exercise

Write a regular expression for the following language

$$\{ w \mid w \text{ has alternating 0's and 1's} \}$$

$(E + F)^*$ ε, $\{x \mid x \in E\}$, $\{y \mid x \in F\}$, $\{xy \mid x \in E \cup F \wedge y \in E \cup F\}$ *

$(a+b)^* = \{\varepsilon, a, b, aa, bb, ab, ba, \cdots\}$

$(1+\varepsilon)(01)^* + (0+\varepsilon)(10)^*$

101

$\left(\boxed{(1+\varepsilon)(01)^*} + \boxed{(0+\varepsilon)(10)^*} \right)^*$

101

$1^*(01)^*$

↳ accepts 1101 not in language.

$$E1 \neq E2$$

$$\boxed{01^* + 1} \qquad \boxed{0(1^* + 1)}$$

1

Justify if $L(E1) = L(E2)$

RE: Operator Precedence

$$\boxed{10^* \text{ vs. } (10)^*}$$

$$\boxed{01^* + 1 \text{ vs. } 0(1^* + 1)}$$

$$\boxed{0 + 1^* \text{ vs. } (0 + 1)^*}$$

DFA: Exercise

0101

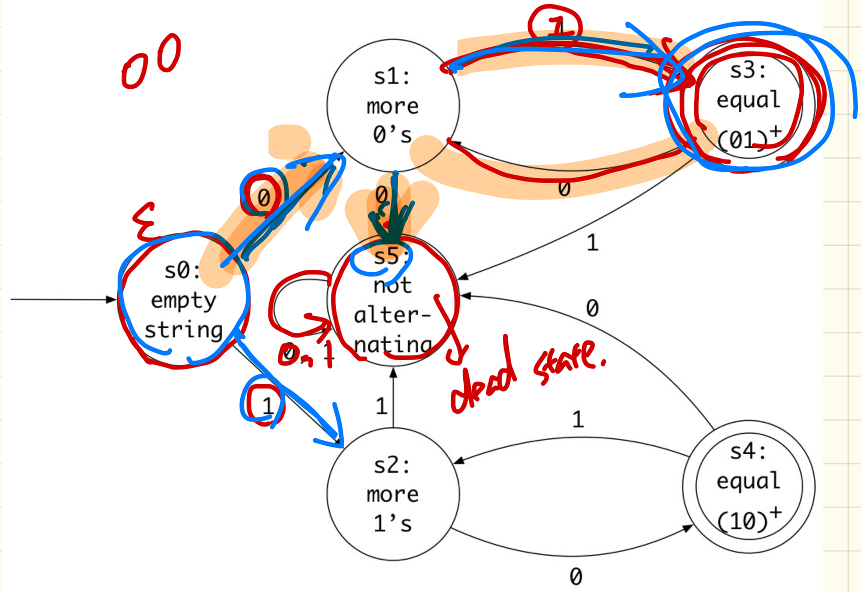
The *transition diagram* below defines a DFA which *accepts* exactly the language

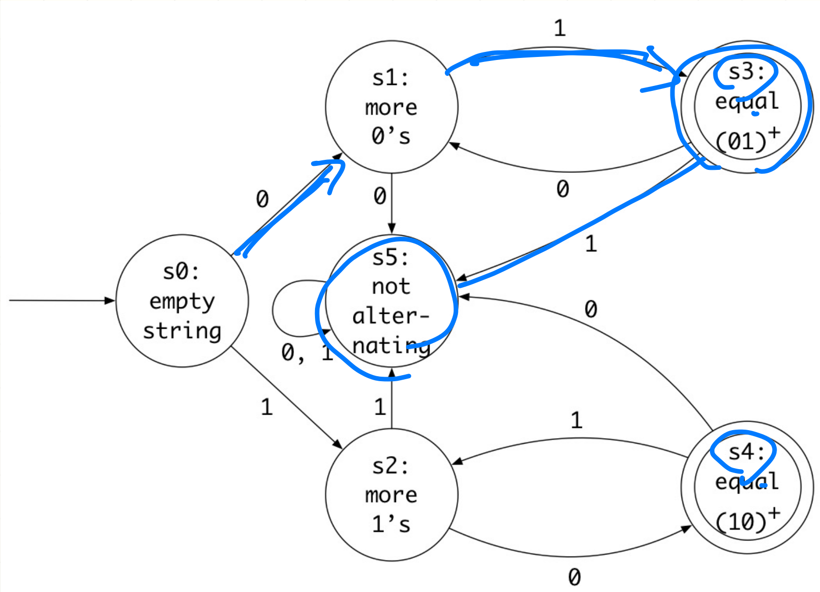
$$\left\{ w \;\middle|\; \begin{array}{l} w \neq \epsilon \\ \wedge \;\; w \texttt{ has equal \# of alternating 0's and 1's} \end{array} \right\}$$

0101 ✓

01010 X

transitions = $|\Sigma_{bin}|$ = |0|1|




s0:
empty
string
s1:
more
0's
s2:
more
1's
s3:
equal
(01)+
s4:
equal
(10)+
s5:
not
alter-
nating
0
1
0, 1

01 ✓

011 ✗

DFA: Formulation (1)

A *deterministic finite automata (DFA)* is a 5-tuple

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

states, alphabet, transition, initial, accepting states

Language of a DFA

$$L(M) = \left\{ a_1 a_2 \dots a_n \;\middle|\; 1 \le i \le n \;\wedge\; a_i \in \Sigma \;\wedge\; \delta(q_{i-1}, a_i) = q_i \;\wedge\; q_n \in F \right\}$$

$Q = \{ s_0, s_1, s_2, s_3, s_4, s_5 \}$

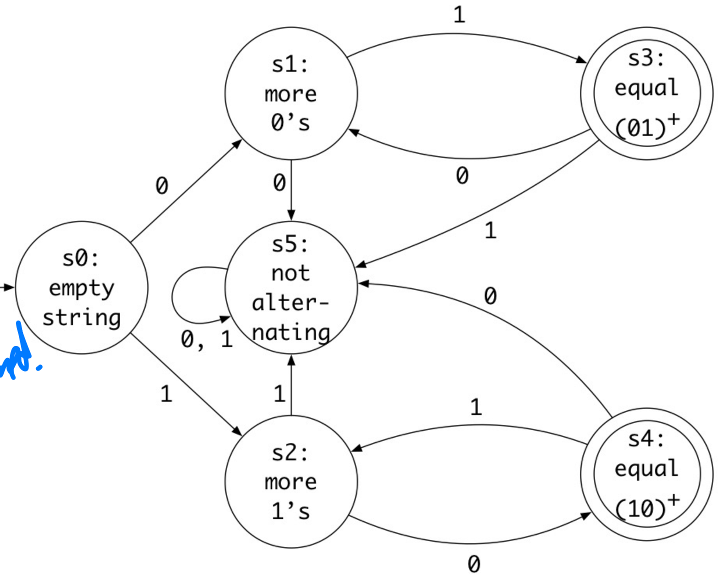
$F = \{ s_3, s_4 \}$

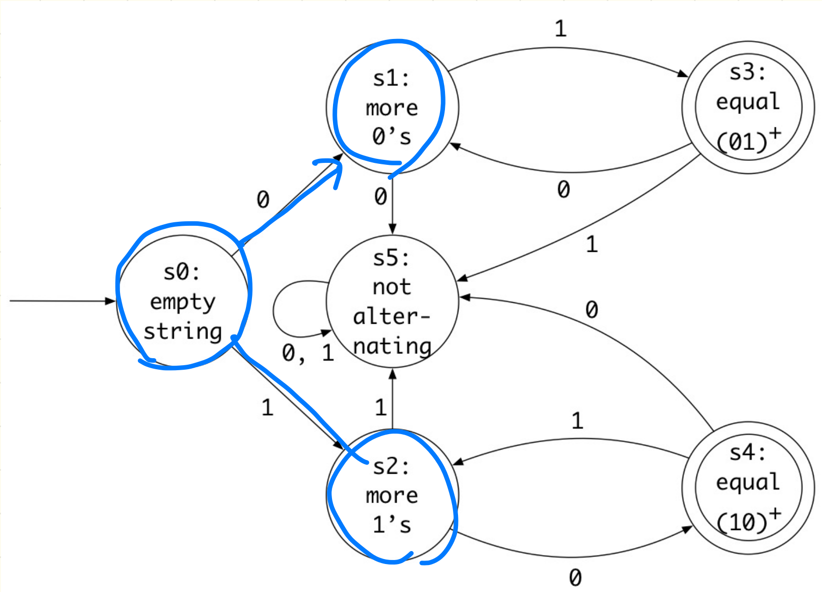
$\rightarrow$ total function: result defined for every domain value

$\not\rightarrow$

$div : Int \times Int \not\rightarrow Int$

$div(3, 0)$ undefined.





δ

C.S. \ Input	0	1
s_0	s_1	s_2
s_1		
s_2		
s_3		
s_4		
s_5		

DFA vs. NFA

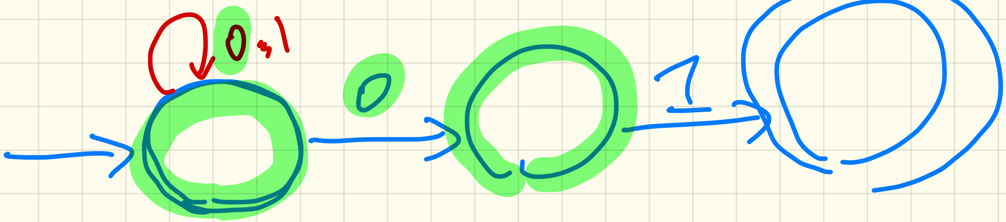
Problem: Design a DFA that accepts the following language:

$$L = \{ x01 \mid x \in \{0,1\}^* \}$$

That is, L is the set of strings of 0s and 1s ending with 01.

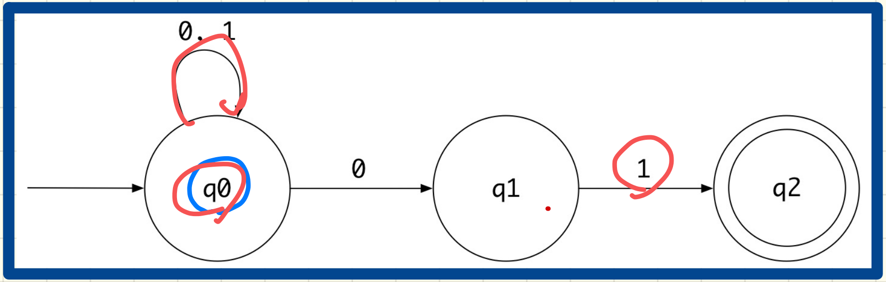
x01

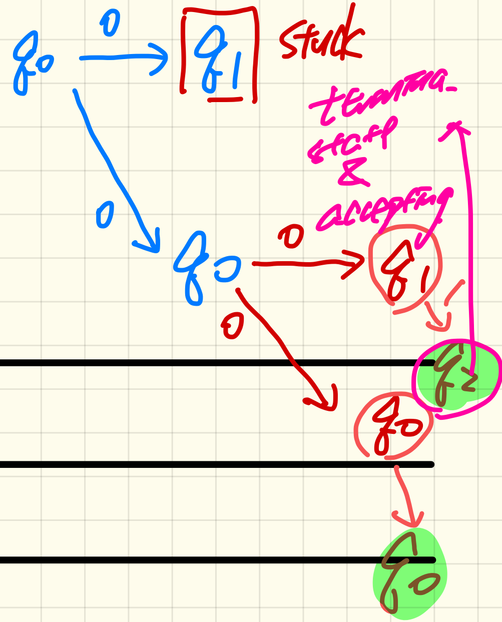
A *non-deterministic finite automata (NFA)* that accepts the same language:



NFA: Processing Strings

How an NFA determines if an input *00101* should be processed:





- **Read 0:**
- **Read 0:**
- **Read 1:**
- **Read 0:**
- **Read 1:**