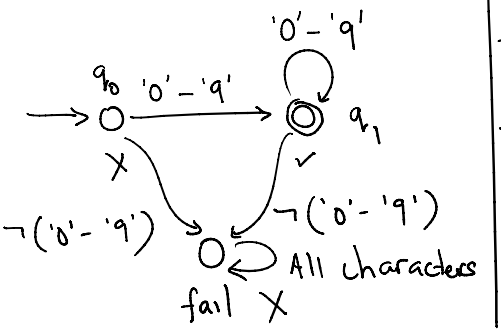
Lecture 16

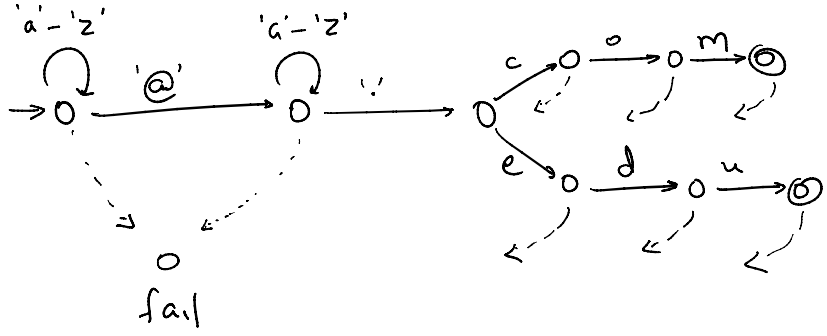
Thursday, October 15, 2020 3:52 PM

- Regex parsing using finite automata.
- CYK parsing (2).
- Type checking



$['0'-'9']^+$

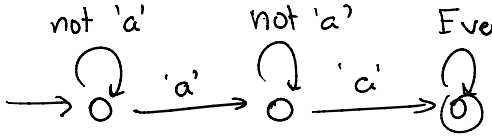
- Initialize state := q_0
- For each character c in the input string:
  - If state = q_0 and c ∈ ['0'-'9'] then: state := q_1
  - Else if state = q_1 and c ∈ ['0'-'9'] then state := q_1
  - Else state := fail.
- Accept iff state = q_1



Email addresses

fail

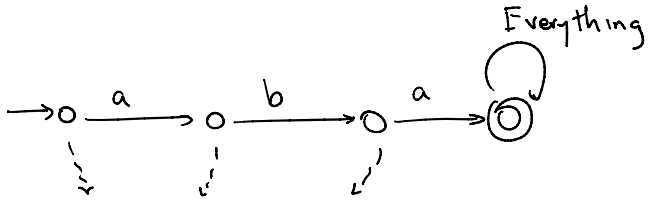
Email addresses



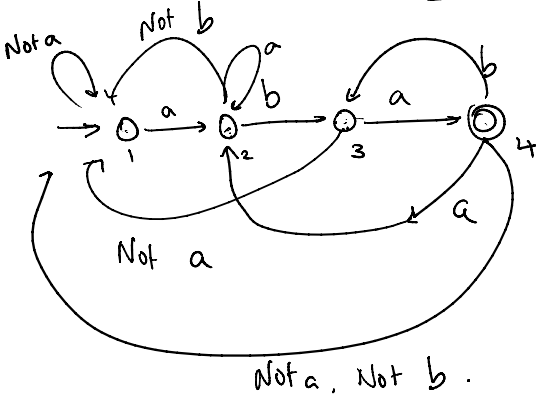
At least two a-s

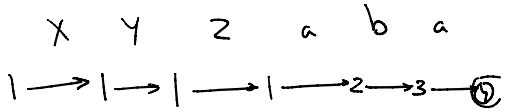
GO fail

"Every string beginning with aba"



Every string ending with aba.



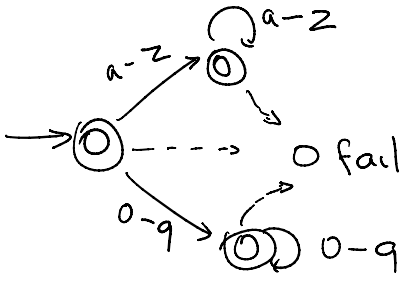


Regular Expressions:

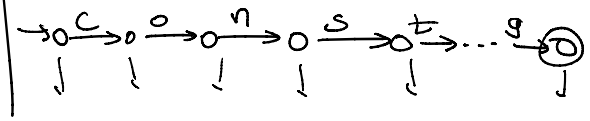
$$r ::= \text{Empty} \mid \text{"Constant String"} \mid r_1 \cdot r_2 \mid r_1 + r_2 \mid r_1^*$$

$| r_1 \cdot r_2 \; | \; r_1 + r_2 \; | \; r_1^*$

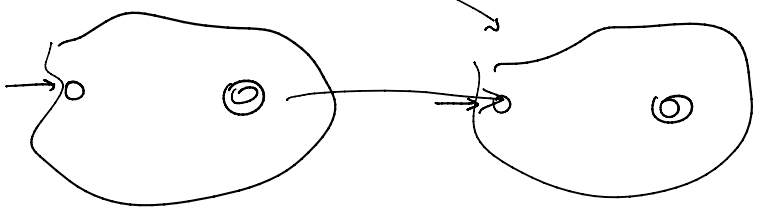
$['a' - 'z']^* + ['0' - '9']^*$



$r ::= Empty$	$r ::= "Constant\ String"$.



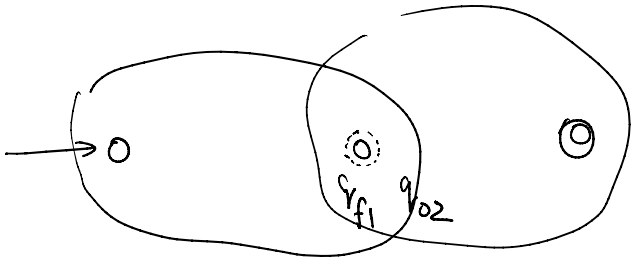
$r = r_1 \cdot r_2$



Proposal 1: For every accepting state of r_1 (q_{f1})

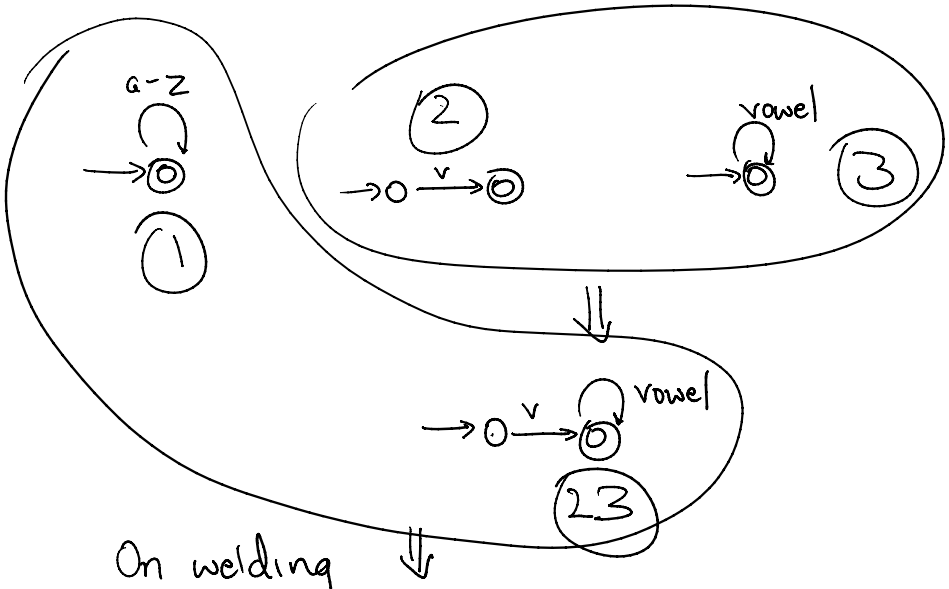
& for every starting state of r_2 (q_{02})

Create a transition $q_{f1} \longrightarrow q_{02}$.

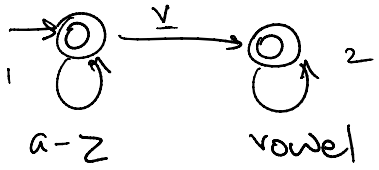


Proposal 2 : Weld the automata together.

$['a'-'z']^* \cdot 'v' \cdot ['a','e','i','o','u']^*$



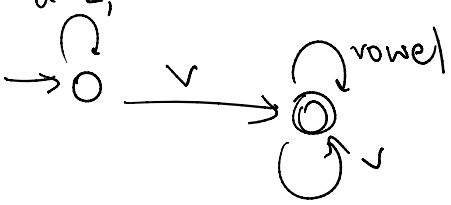
On welding



For each character c

- If curr state = 1 & character = v :

  next state = ____



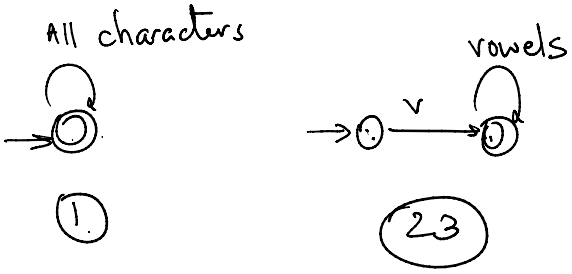
Works, but needed

Question 1 : Exactly what does the code do on seeing v?

Works, but needed human insight

does the word start seeing v?

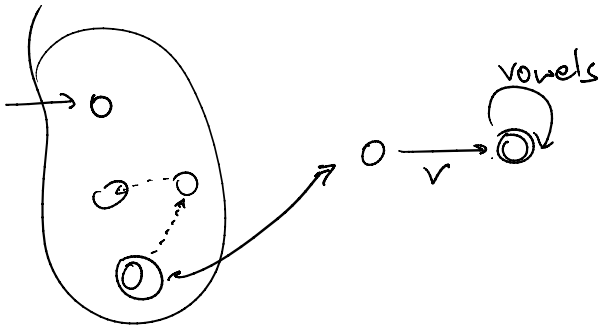
Question 2:



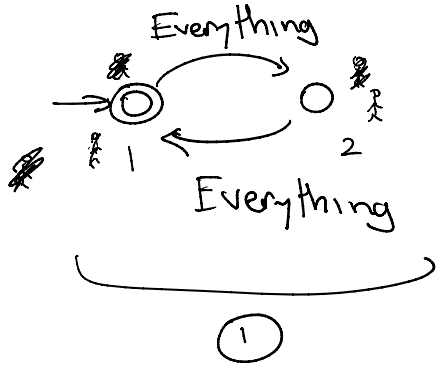
abvbveea

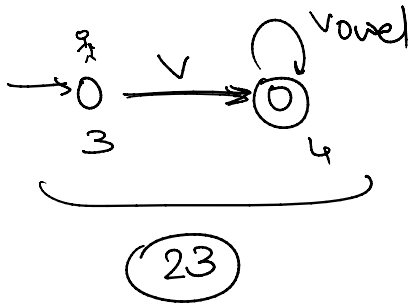
Transition to machine (23) or stay in machine (1)

Transition to machine (23) or stay in machine (1).



(Even number of chars) . v . (vowel)*





abbavabbavaavee1

1.1 → 23.4

Do we jump back to 1.1 or

① · i → ㉓ · 4 back to ① · 1 or
do we jump back to
① · 2 ?

Every time machine ① reaches an accepting state,

- split the stick figure into two.
  First piece continues to explore machine ①
  The second piece moves into machine ㉓.

If you have k states, you can never have more than k stick figures

- For each character c
  - For each stick figure
    - One giant switch case:
      If in state q & $c = a$, then move to q'.

$O(k)$ time

$k \simeq 15$

$15n$

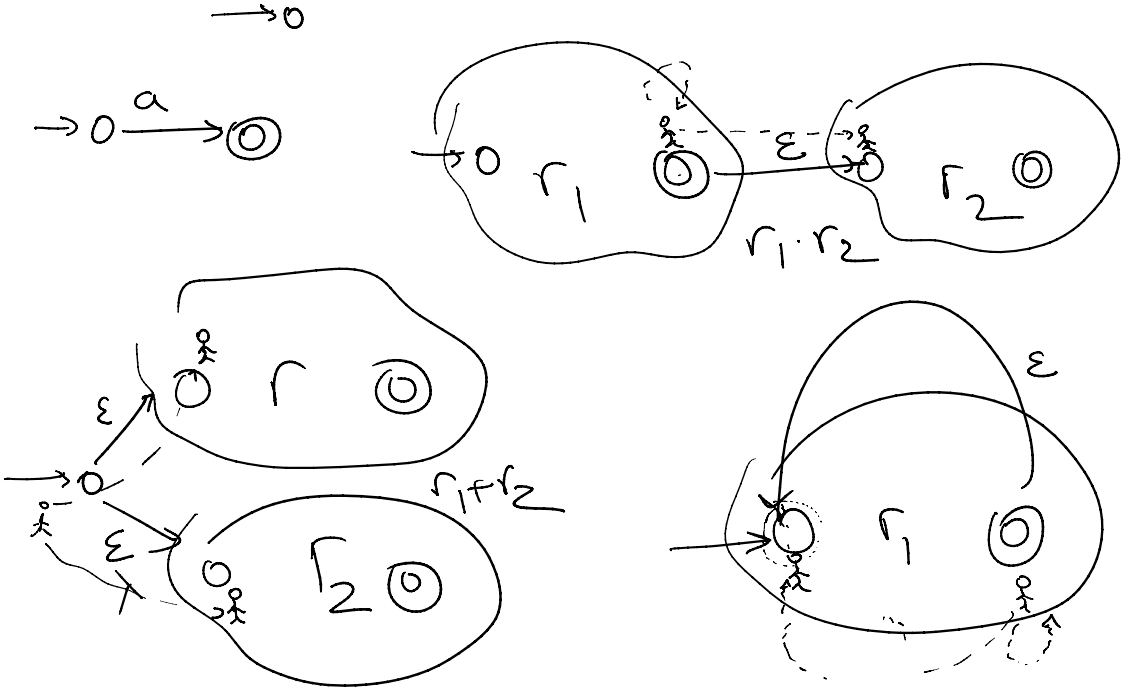
$O(nk)$

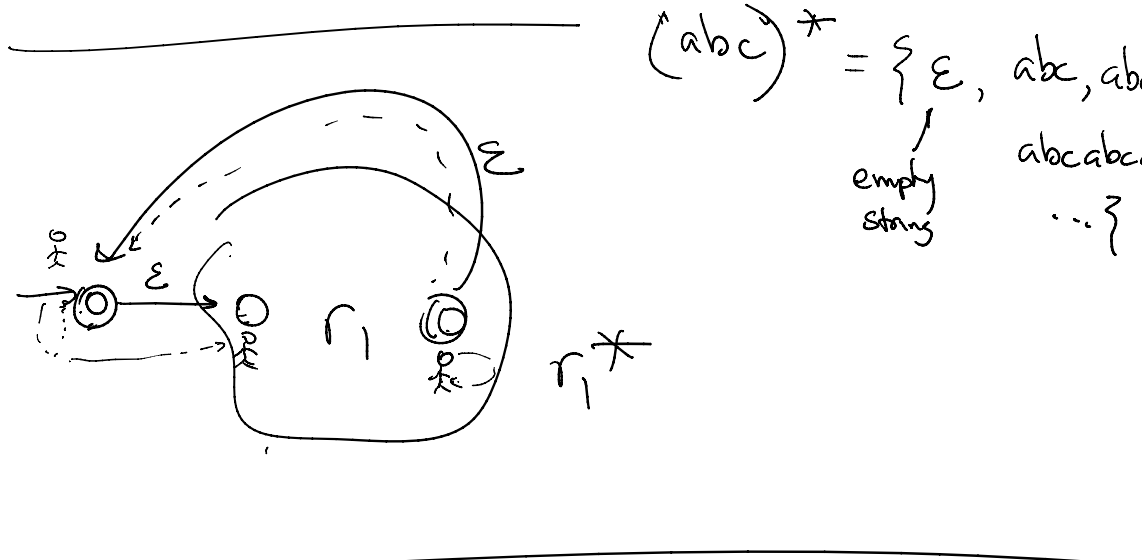
Long string of length n.

string of length n.

$O(nk)$ — Assume k is constant.

$r ::= \text{Empty} \mid \text{"a"} \mid r_1 \cdot r_2 \mid r_1 + r_2 \mid r_1^*$





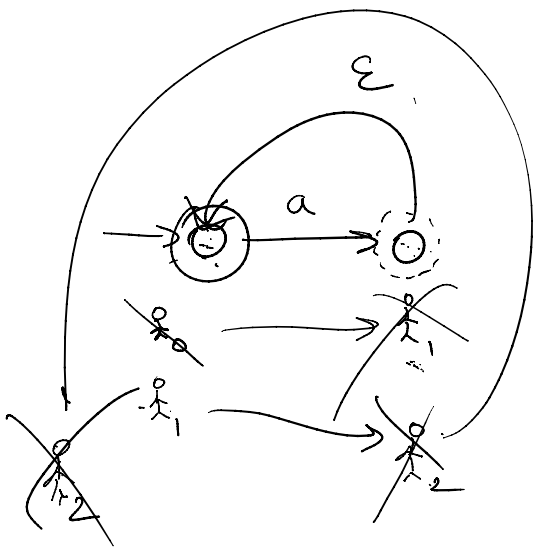
$(abc)^* = \{\varepsilon, abc, abcabc, abcabcabc, \ldots\}$

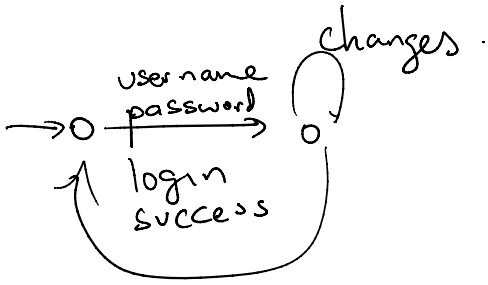
(ε: empty string)

Example: a^*

a a b a

0 1 2 3 4





reactive systems.

If you have k states in the NFA (board game)

then each state can either be occupied with a stick figure or not.

So your board can be in 2^k different configurations.

If k is finite then so is 2^k.

Replace the word "configuration" with "state".

We argued that NFAs & DFAs are expressively equivalent.

This argument is called the subset construction.

CYK algorithm for parsing CFGs:

$O(n^3 |G|)$ time.

$n^{2.38}$

LR(1) parsers Deterministic CFLs.

How types work?