Finite automata & Morphological parsing

→ useful for compact representation of lexical rules (singular/plural, prefix suffix, finding root of a word), or idioms

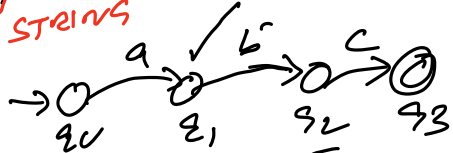
↓ group of words whose meaning is not deducible from individual words. e.g. "India Book house", "Indian sweets", ...

To parse (a word):

books	book + PLT N
likely	Like + Adv
.....	

I/P = STRING ✓ — O/P = STRING ✓

FA:



= CLC

I/P = STRING
O/P = Y/N

$q_3 \in F$, FA machine $M = (Q, \Sigma, q_0, \delta, F)$

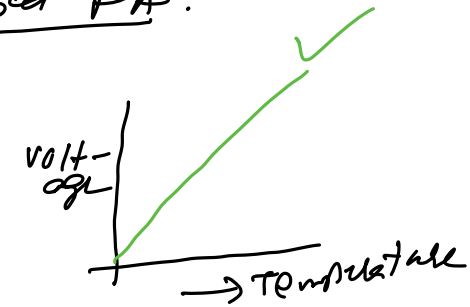
$\delta : Q \times \Sigma \to Q$

We are interested in efficiency: — Time & Space ← Size of I/P
↑ No. of Transitions

FA: O/P ⇒ Y/N.

To use FA that reads i/p as strings and also writes o/p as strings, then should be two tapes one for i/p & one for o/p. So this is a **modified FA**.

Example: Temperature sensor (Transducer)



so the above FA with i/p as string & o/p also as string is called **FST** (Finite state Transducer)

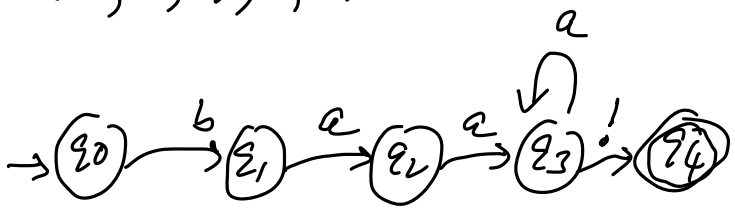
↓

has applications: for representation of large dictionaries, Computational morphology local grammar, syntax.

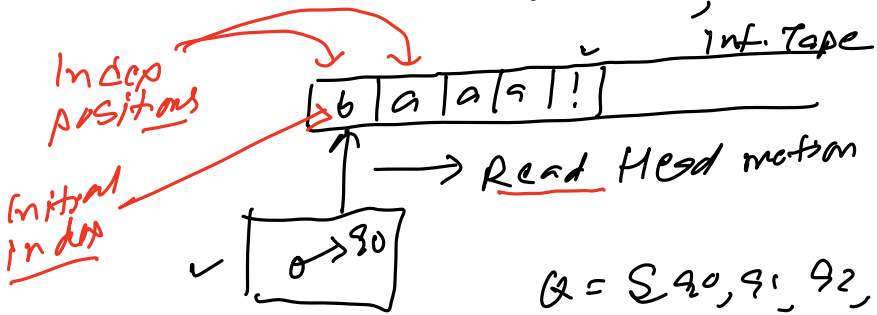
REGE := a^*, R. Lang = { a, aa, aaa, ... }

FA ∴ F.A ↔ R. Lang ↔ Reg Ex. ↔ FA

Let there is FA $M = (Q, \Sigma, q_0, \delta, F)$



Its language $= \{ baa!, baaa!, baaaa!, \ldots\ldots \}$



$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$F = \{q_4\}$, $\Sigma = \{a, b, !\}$

$\delta =$

State	I/P a	b	!
q_0	–	q_1	–
q_1	q_2	–	–
q_2	q_3	–	–
q_3	q_3	–	–
q_4	–	–	–

"–" = empty state

I/P :- ab – rejected
abb – "
baa! Accepted
baaa! "

Algorithm dfa-recognize (tape, machine) return accept/reject.

function

```
1.  index ← initial position on tape
2.  While True do
3.      If end-of-i/p then
4.          if current-state = accept then
5              return accept
6.         else
7.             return reject
8          endif
9.     else
10         if transition-table [current-state, tape[index]] ==
                                                    empty then
11.            return reject
12         else
13 ↓       current-state ← trans-table [current-state, tape[index]]
```

ALSO OF FA

14. index++;

15 endif

16. endif

17. end while