CSE 322, Fall 2010

Regular Expressions

Regular expressions over Σ

ϕ is an r.e.

ε .... r.e.

a ..... for each $a \in \Sigma$

if R_1 & R_2 are r.e.s,
then so are

$(R_1 \cup R_2)$

$(R_1 \circ R_2)$

$(R_1{}^*)$

---

the language denoted by R, $L(R)$
is :

$L(\phi) = \phi$

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

$\vdots$

$L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$

$L(\underbrace{(\phi^*)}_{\text{R.E.}}) = L(\phi)^*$

$= \phi^*$

$= \{\varepsilon\}$

Short hands

$\Sigma = \{a, b, c\}$

$L(((a \cup b) \cup c)) = \Sigma$

$(\underline{\Sigma}^* \cup \varepsilon) \cdot a$

$(((\underline{(a \cup b) \cup c})^* \cup \varepsilon) \cdot a)$

precedence & associativity

$(a \cup b \cup c)$

$a \cup b \cdot c^*$

$(a \cup (b \cdot (c^*)))$

"words ending with ".TXT" "

$\Sigma^* . TXT$

$(a \cup b \cup \cdots \cup z) \cdot (a \cup \cdots \cup z \cup 0 \cdots 9)^*$

$l \cdot (l \cup d)^*$

Shorthand

$\left.\Sigma\Sigma^*\right\}\ \Sigma^+$

$\left.aa^*\right\}\ a^+$

$(\Sigma\Sigma)^*$

$0^* 1 0^*$

$(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)$

$\Sigma\Sigma$

$00 \in \quad 0^*(10^*10^*)^*$

$00 \notin \quad (0^*10^*10^*)^*$

$(0^*10^*1)^*0^*$

$(d^* . d^+ \cup d^+ . d^*)(\varepsilon \cup E(\varepsilon \cup + \cup -)d^+)$

Theorem:

∀ regular expression R ∃ an NFA M_R st $L(R) = L(M_R)$

Proof:

By induction on K, the # of $\cup, \circ, *$ operators in R

Base cases ($K=0$):

Then R is "ϕ", "ε", or "a" for $a \in \Sigma$.
Explicitly give simple NFA's recognizing ϕ, $\{\varepsilon\}$, and $\{a\}$ for each $a \in \Sigma$.
(details omitted)

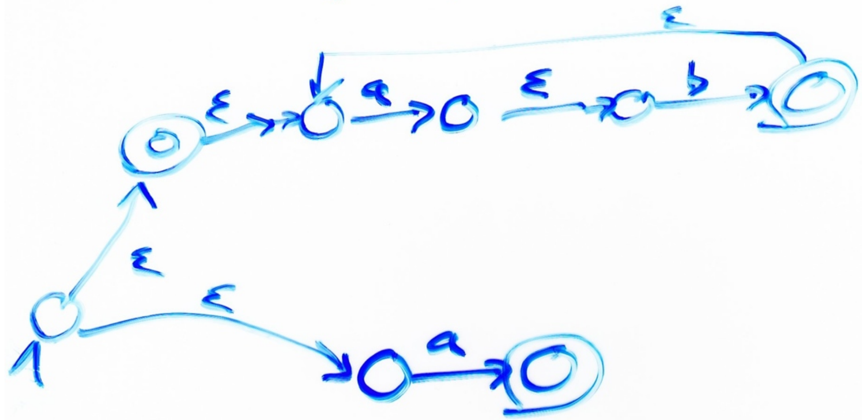
Induction Step (R has $K > 0$ operators)

$\boxed{\text{I.H.: assume that for all regular expressions } R' \text{ with } < K \text{ operators, } \exists \text{ NFA } M_{R'} \text{ recognizing } L(R')}$

R has $K > 0$ operators. So R is $(R_1 \cup R_2)$ or $(R_1 \circ R_2)$ or $(R_1)^*$ where R_1 (& R_2 if any) have $\leq K-1$ operators. By I.H., $\exists M_{R_1}$ (& M_{R_2}) st. $L(R_i) = L(M_{R_i})$, $i=1,2$. Modify/join it/them as in previous proofs of closure under $\cup, \circ, *$ to get M_R st. $L(R) = L(M_R)$.

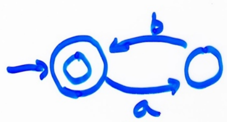
Example

$(ab)^* \cup a$

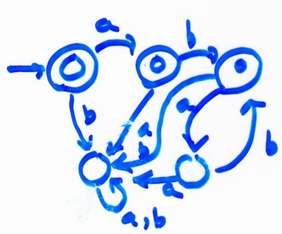


Converse?

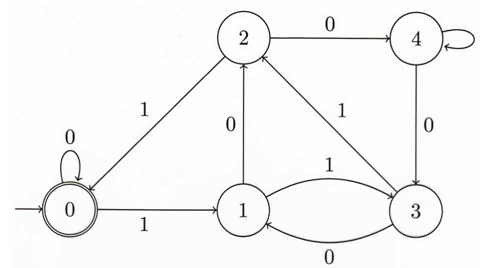
For every D/NFA ∃ reg expr defining the same language.



→ $(ab)^*$



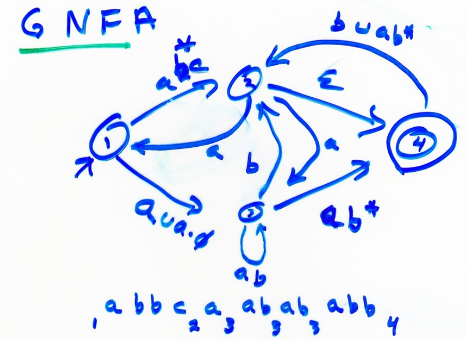
$\xrightarrow{?}$



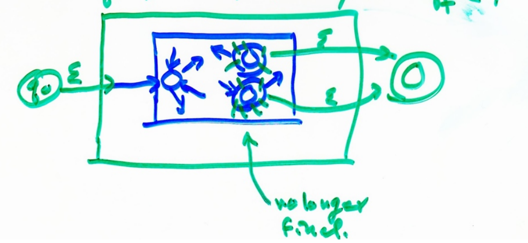
pattern? $\left\{ \begin{array}{r} 0 \\ 101 \\ 1010 \\ 1111 \\ 10100 \\ 11001 \\ 11110 \\ 100011 \\ 101000 \\ 101101 \\ 110010 \\ 110111 \\ 111100 \\ \vdots \end{array} \right.$

Every regular language can be described by a regular expression

GNFA



Note: No loss in assuming no edges into q_0 / out of F / only one $q_f \in F$



GNFA

$G = (Q, \Sigma, \delta, q_0, q_f)$

$Q, \Sigma, q_0, q_f \in Q$ as usual

$\delta : (Q - \{q_f\}) \times (Q - \{q_0\}) \to R_\Sigma$

Regular expressions over Σ

Defn

- G can be in state $q \in Q$ after reading $x \in \Sigma^*$ if $\exists k \geq 0$,
  $\exists\ r_0, r_1, \dots, r_k \in Q$
  $\exists\ x_1, \dots, x_k \in \Sigma^*$
  such that
  (i) $x = x_1 \cdot x_2 \cdot \dots \cdot x_k$
  (ii) $r_0 = q_0$
  (ii) $r_k = q$
  (iii) $\forall\, 1 \leq i \leq k,\ x_i \in L(\delta(r_{i-1}, r_i))$
- $L(G) = \{ x \mid$ G can be in state $q_f \dots \}$

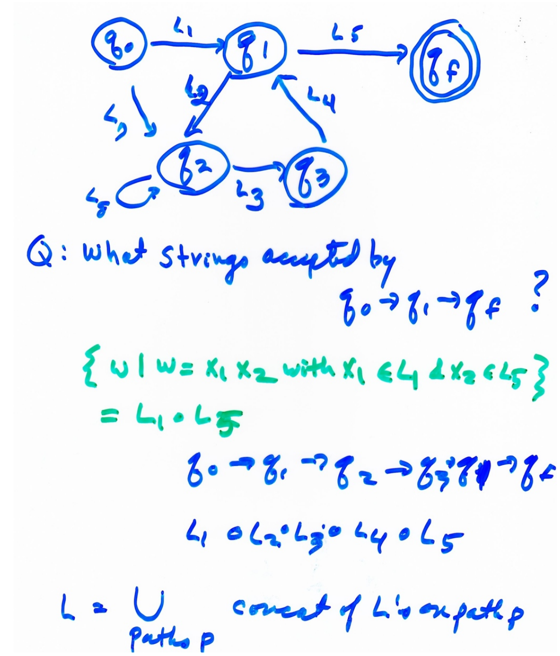
Note: δ syntax a little different; maps state pair to label (reg. exp.) rather than state $\times$ symbol $\to$ new state.

Theorem

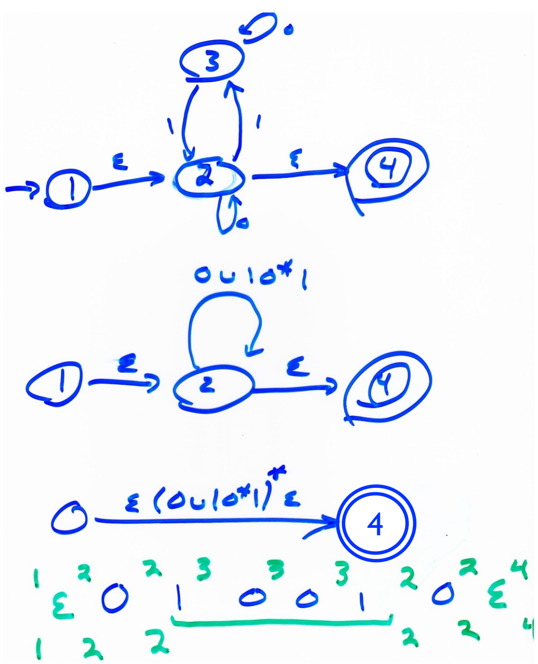
If L is accepted by a GNFA, then L is regular

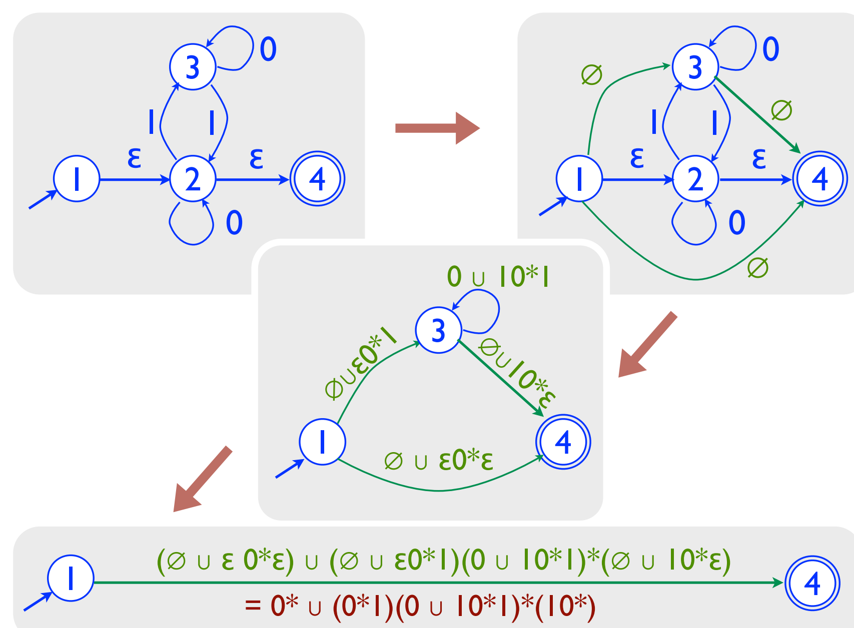
Pf sketch: replace edge labeled "r" by NFA equivalent to r based on previous theorem.

If L is regular, then L=L(R) for some regular expression R



Proof will take FA for L, & reduce it to a (G)NFA for same L with progressively fewer states until R becomes obvious.





Given GNFA $G = (Q, \Sigma, \delta, q_0, q_f)$ with > 2 state } "the old machine"

Notation $\forall q_i \neq q_f,\ q_j \neq q_0$

$$\boxed{r_{ij} = \delta(q_i, q_j)}$$

Pick any state $q_k \neq q_0, q_f$

Build GNFA $G' = (Q', \Sigma, \delta', q_0, q_f)$ with one less state as follows: } "the new machine"

$$Q' = Q - \{q_k\}$$

$$\boxed{\delta'(q_i, q_j) = r'_{ij} = r_{ij} \cup r_{ik} r_{kk}^* r_{kj}}$$

Claim 1 G & G' are equivalent

To prove this, it is useful to focus on a sub problem: how do edges in G' relate to paths in G?

In a nutshell, delete state k from G, but enlarge language on each edge to compensate, so that potential contribution of k is added to each edge in G'

```
               r   s   u   v   w   x   y   z
Path in G:   1   2   3   k   4   k   k   2   1 ...

Path in G': 1   2   3       4           2   1 ...
               r   s    uv       wxy        z
```

strings in L(edge reg exp)

Relating edges of G' to paths of G

A path in G: any sequence of states

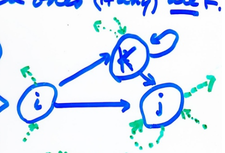
A simple path in G: any sequence of ≥ 2 states s.t. 1st & last are not k, and all intermediate ones (if any) are k.

$i \to j$

$i \to k \to j$

$i \to k \to k \to j$

$\vdots$



Other edges ignored/forbidden

The Point:

(a) every path in G can be decomposed into simple paths

(b) every edge in G', say $i \to j$, corresponds to the set of all simple paths in G with those endpoints

Claim 2

$L(r'_{ij}) = \{ w \mid$ G can move from i to j reading w and passing through no intermediate states except possibly k. $\}$

Equivalently:

$L(r'_{ij}) = \{ w \mid$ G can move from i to j reading w along a simple path $\}$

$$= L(r_{ij} \cup r_{ik} \cdot r_{kk}^* \cdot r_{kj})$$

Claim 4 $\forall$ NFA $\exists$ equiv. reg. expr.

Proof: NFA $\to$ GNFA $\to$ 2-state GNFA $\to$ r.e.

by induction on k, using claim 1

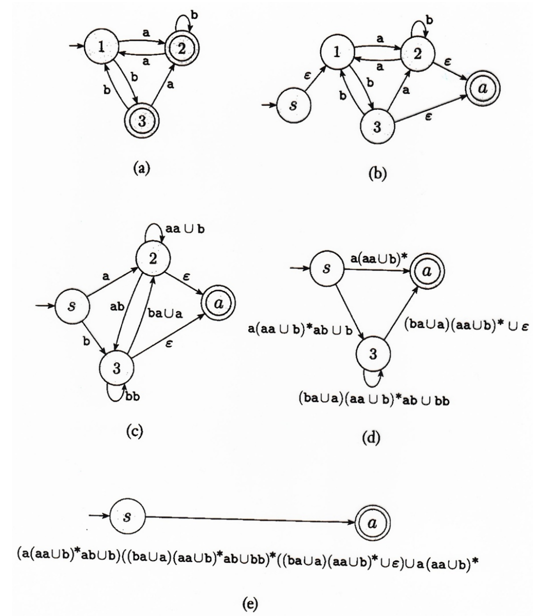


FIGURE 1.69

Summary

L is regular $\Leftrightarrow$

L = L(M) for some DFA M

$\Leftrightarrow$ L = L(N) .... NFA N

$\Leftrightarrow$ L = L(G) ..... GNFA G

$\Leftrightarrow$ L = L(R) .... reg. exp. R