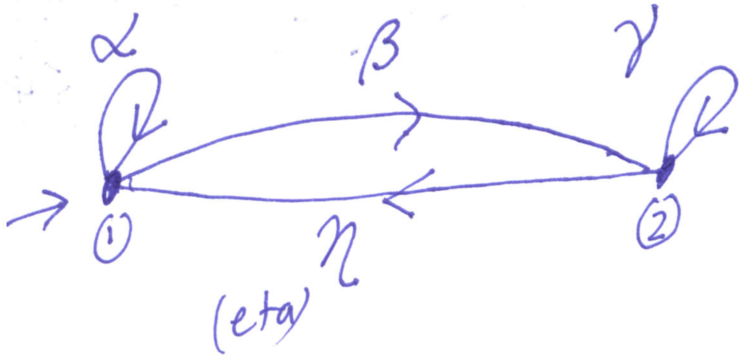
The General 2-State GNFA

$L_{pq} = \{x \in \Sigma^*:$ the FA can process x from p to $q\}$.



$$L_{11} = (\alpha \cup \beta\gamma^*\eta)^*$$

$$L_{22} = (\gamma \cup \eta\alpha^*\beta)^*$$

$\alpha, \beta, \gamma, \eta \in \mathrm{Regexp}(\Sigma)$

$$L_{12} = L_{11} \cdot \underbrace{\beta\gamma^*}_{\text{going to ② on the "last lap" without coming back to ①}}$$

Stephen Kleene

$$= (\alpha \cup \beta\gamma^*\eta)^*\beta\gamma^*$$

ie: If both ①,② $\in F$, $s =$ ①:

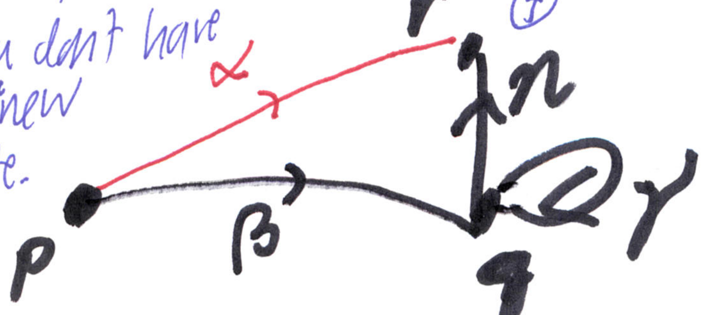
$. = \underline{L_{11} \cup L_{12}}$

$$L_{12} = \alpha^*\beta \cdot L_{22} = \alpha^*\beta(\gamma \cup \eta\alpha^*\beta)^*$$

$$L_{21} = L_{22} \cdot \eta\alpha^* = \gamma^*\eta \cdot L_{11} = (\gamma + \eta\alpha^*\beta)^*\eta\alpha^* = \gamma^*\eta(\alpha + \beta\gamma^*\eta)^*$$

⇒ More Efficient FA → Regexp Conversion (especially when at most 1 state besides ⓢ is in F)

⇑ ...hen you don't have ...add a new ...nal state.



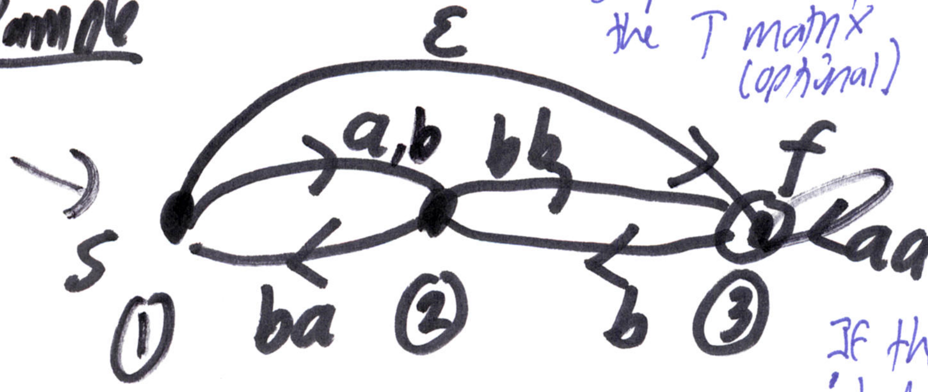
$T(p,r) \mathrel{+}= \beta\gamma^*\eta$

$so = \alpha + \beta\gamma^*\eta$

1. For each $q \notin F$, $q \neq s$: eliminate q by bypassing each incoming edge (p, β, q) to each outgoing arc (q, η, r).

Example

Step 0: Initialize the T matrix (optional)



T_0:

	1	2	3
1	$\emptyset$	$a+b$	ε
2	ba	$\emptyset$	bb
3	$\emptyset$	b	aa

If there is no loop at a state p, it doesn't matter whether you write $T(p,p)=\emptyset$ or $T(p,p)=\varepsilon$. (ultimately because $\emptyset^* = \varepsilon^* = \varepsilon$).

Option Could renumber so that $f=2$ and nonaccepting states are 3 to n.

Elim 2

Incoming ~~(S, a, 2)~~ ~~(S, b, 2)~~ Outgoing

$$\boxed{T(p,r)_{new} = T(p,r)_{old} + T(p,q)\,T(q,q)^*\,T(q,r)}$$

$(1, a+b, 2)$ $(2, ba, 1)$

$(3, b, 2)$ $(2, bb, 3)$

$\therefore$ Must update all of $T(1,1)$, $T(1,3)$, $T(3,1)$ and $T(3,3)$ $\boxed{\emptyset^* = \varepsilon}$

$$T(1,1)_{new} = T(1,1)_{old} + T(1,2)\cdot T(2,2)^*\cdot T(2,1)$$

OK since "Tpp"

$$= \emptyset + (a+b)\cdot\varepsilon\cdot ba = (a+b)ba$$

$$T(1,3)_{new} = T(1,3)_{old} + T(1,2)\,T(2,2)^*\,\underline{T(2,3)}$$

cannot forget $p \neq q$ →

$$= \varepsilon + (a+b)\cdot\varepsilon\cdot\underline{bb} = \varepsilon + (a+b)bb.$$

$$T(3,1)_{new} = T(3,1)_{old} + T(3,2)\,T(2,2)^*\,T(2,1)$$

$$= \emptyset + b\cdot\varepsilon\cdot ba = bba$$

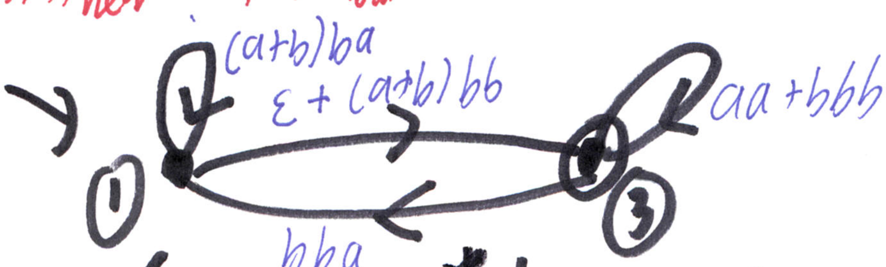
$$T(3,3)_{new} = T(3,3)_{old} + T(3,2)\,T(2,2)^*\,T(2,3) = aa+bbb$$

$T' =$

$(a+b)ba$	$a+b$	$\varepsilon+(a+b)bb$
~~ba~~	~~$\emptyset$~~	~~bb~~
bba		$aa+bbb$

$T' =$

	1	3
1	$(a+b)ba$	$\varepsilon+(a+b)bb$
3	bba	$aa+bbb$



$$L = T'_{13} = T'(1,1)^*\cdot T'(1,3)\cdot\big(T'(3,3) + T'(3,1)\,T'(1,1)^*\,T'(1,3)\big)^*.$$

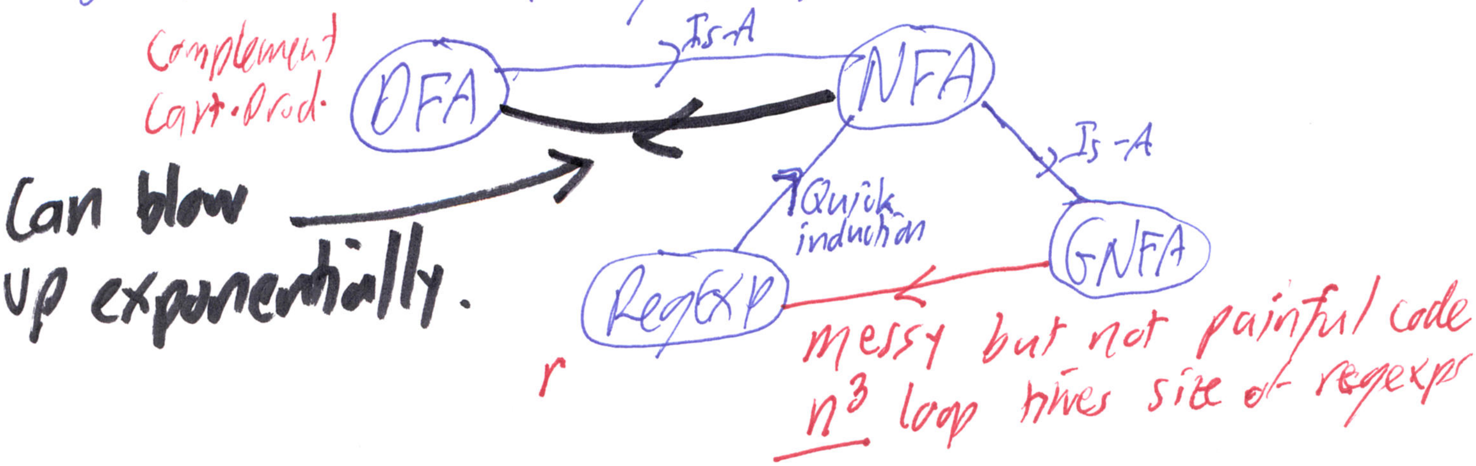
$s' \quad = ((a \cup b)ba)^* (\varepsilon \cup (a \cup b)bb)[(aa \cup bbb) \cup bba((a \cup b)ba)^*(\varepsilon \cup (a \cup b)bb)]^* \quad f'$ ③

Kleene's Theorem of Regular Languages:

For any language L over an alphabet Σ (i.e. $L \subseteq \Sigma^*$), the following statements are equivalent:

1. There is a DFA M such that $L = L(M)$
2. There is an NFA N such that $L = L(N)$
3. There is an r in Regexp(Σ) such that $L = L(r)$.

Any one can be "the" defn of "L is regular."



<u>Corollary</u>: For every regular expression r there is a regular expression r' such that $L(r') = \sim L(r)$.

<u>Proof By Process</u>: Given r,

1. Build NFA N_r s.t. $L(N_r) = L(r)$.
2. Convert N_r to a DFA M_r s.t. $L(M_r) = L(N_r)$ ⚠
3. Complement M_r to M'_r s.t. $L(M'_r) = \sim L(M_r)$.
4. Convert M'_r into r' s.t. $L(r') = L(M'_r)$. ☒

In Practice, a ~ operator in UNIX "egrep" etc. is allowed only at bottom level nesting, eg on <u>chars</u>