The General $2$-State GRFA

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

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

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

$L_{12} = L_{11} \cdot \beta \gamma^*$

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

1. For each $q \in F$ such that $q \notin S$:
   - eliminate $q$ by bypassing each incoming edge $(p, \beta, q)$ to each outgoing arc $(q, \eta, r)$.
**Example**

Step 0: Initialize the $T$ matrix (optimal)

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>$\emptyset$</td>
<td>$a+b$</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>(2)</td>
<td>$ba$</td>
<td>$\emptyset$</td>
<td>$bb$</td>
</tr>
<tr>
<td>(3)</td>
<td>$\emptyset$</td>
<td>$b$</td>
<td>$aa$</td>
</tr>
</tbody>
</table>

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

**Elim**

Incoming $(s,a+2)$

- $(s,b,2)$

Outgoing

- $T(1,1)$
- $T(1,3)$
- $T(3,1)$
- $T(3,3)$

$\emptyset^* = \varepsilon$

$T(1,1)_{new} = T(1,1)_{old} + (a+b) \cdot \varepsilon \cdot ba = (a+b)ba$

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

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

- $T(3,3)_{new} = T(3,3)_{old} + (a+b)ba$

$L = T_{13} = T'(1,1).T'(1,3).T'(3,3) + T'(3,1)T'(1,1).T'(1,3)$.
Kleene's Theorem of Regular Languages:

For any language \( L \) over an alphabet \( \Sigma \) (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 a \( r \) in \( \text{Regexp}(\Sigma) \) such that \( L = L(r) \).

**Complement**

- DFA \( \rightarrow \) NFA
- Complement
- DFA

**Can blow up exponentially**.

**Corollary**: For every regular expression \( r \) then is a regular expression \( r' \) such that \( L(r') = \overline{L(r)} \).

**Proof By Process**:

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. \( L(M'_r) = \overline{L(M_r)} \).
4. Convert \( M'_r \) into \( r' \) s.t. \( L(r') = \overline{L(r)} \).

In practice, a \( \sim \) operator in UNIX "grep" etc. is allowed only at bottom level nesting, e.g., on "chmod"