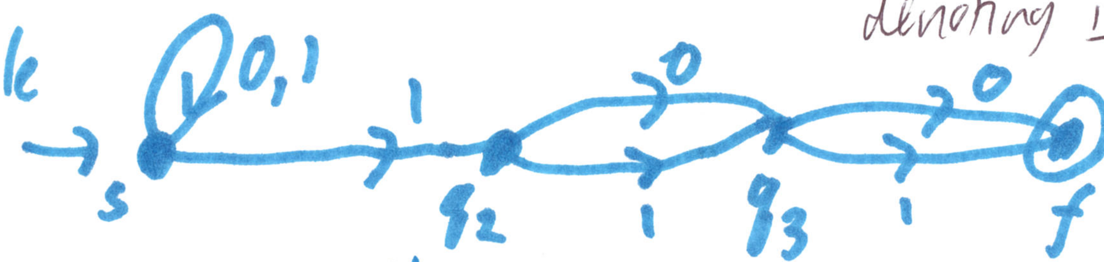
An NFA is a 5-tuple $N = (Q, \Sigma, \delta, s, F)$ where Q, Σ, s, F are as before and:

$$\delta \subseteq Q \times \Sigma \times Q$$

I.e. δ is a set of triples (p, c, q) $p, q \in Q$, $c \in \Sigma$ denoting <u>instructions</u>.

Example N_3:



$L(N_3) = \{x \in \{0,1\}^* :$ the 3rd char from the right is a 1

A DFA "Is-A" NFA in which the δ <u>relation</u> is actually a function from $Q \times \Sigma$ to Q.

<u>Def^n</u>: An NFA can process a string x <u>from</u> state p to state q ($q = p$ allowed) if there is a <u>computation</u>

$$(q_0, x_1, \overbrace{q_1)\,(q_1}, x_2, q_2) \cdots (q_{n-2}, x_{n-1}, \overbrace{q_{n-1})\,(q_{n-1}}, x_n, q_n$$

which is a sequence of instructions that "match like dominoes" where $q_0 = p$, $q_n = q$, and $x = x_1 \cdots x_n$.

If $x = \varepsilon$, then $p = q = s$.

<u>Def^n</u>: An <u>NFA_ε</u> also allows instructions (p, ε, q). The processing def^n now has $(q_0, u_1, q_1) \cdots (q_{t-1}, u_t, q_t)$ where $x = u_1 \cdots u_t$ and $t \geq n$ can happen when some $u_i = \varepsilon$

Example: N_3 *can* process $X = 10\underline{1}1$ from s to q_3. Let's try some computation traces

"Too Eager":

$(s,1,q_2)(q_2,0,q_3)$ must next do $(q_3,1,f)$ but cannot process the final 1 in X

"Too Timid":

$(s,1,s)(s,0,s)(s,1,s)(s,1,q_2)$ but falls short of $q = q_3$.

"Just Right":

$(s,1,s),(s,0,s)(s,1,q_2)(q_2,1,q_3)$. ✓

Defn: Given N, $\underline{\underline{L_{p,q}}} = \{x \in \Sigma^*: N \text{ can process } x \text{ from } p \text{ to } q\}$

and now formally, $\underline{L(N)} = \underline{\bigcup_{q \in F} L_{s,q}}$.

Note how the interpretation is "benefit of doubt" $1011 \in L_{sq_3}$ even though "your odds are 1-in-3 or 1-in-4".

But $1011 \notin L_{sf}$, so not in $L(N)$, because there is no way to process it all from s to f.

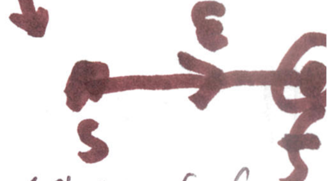
Regular Expression: $L(N_3) = (0+1)^* \cdot 1 \cdot (0+1)^{2}$.

Def^n and Theorem with <u>Reg Exps</u>, by Induction

<u>Basis</u>:

$\underset{\sim}{\emptyset}$ is a regexp, $L(\underset{\sim}{\emptyset}) = \emptyset$, and $N_\emptyset =$

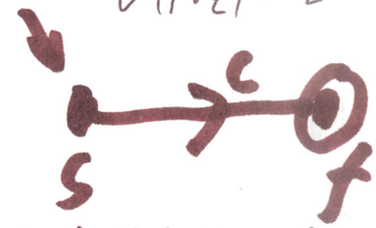
$\underset{\sim}{\varepsilon}$ is a regexp, $L(\underset{\sim}{\varepsilon}) = \{\varepsilon\}$, and $N_\varepsilon =$



Now N_ε can process ε from s to f, so $\varepsilon \in L(N_\varepsilon)$, indeed $L(N_\varepsilon) = \{\varepsilon\}$

For any char $c \in \Sigma$,

$\underset{\sim}{c}$ is a regexp, $L(\underset{\sim}{c}) = \{c\}$, and $N_c =$



= The <u>Language</u> (= set<string>)
whose only member is the <u>string</u> "c"
whose only entry is the <u>char</u> c.
(string = <u>list</u><char>)

Now N_c cannot anymore process ε from s to f, but can process (only) the string c

<u>Induction</u>: Let any two regexps α and β be given

Then

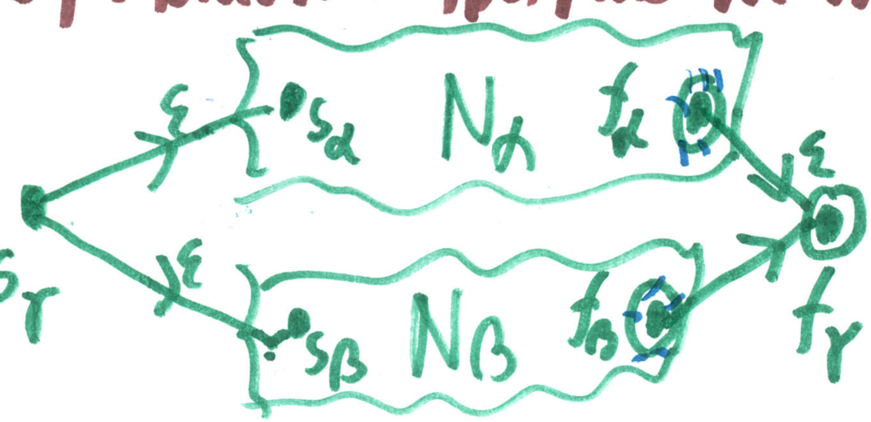
other sources use $\cup$ or $|$

- $\gamma = \alpha + \beta$ is a regexp, $L(\gamma) = L(\alpha) \cup L(\beta)$

and by inductive hypothesis we may <u>take</u> NFAs N_α and N_β st $L(N_\alpha) = L(\alpha)$ and $L(N_\beta) = L(\beta)$.

Build:

$N_\gamma =$



To Be Continued ...