A nondeterministic Finite automaton (NFA) is a 5-tuple $(S, s_0, \Sigma, R, F) = N$ where

as for FSA:
- S = a finite state set, s_0 is the start state $\in S$,
- Σ is the input alphabet, and $F \subseteq S$ the accept states

and $R \subseteq S \times (\Sigma \cup \{\lambda\}) \times S$

i.e. a set of (current state, input consumed, next state) triples

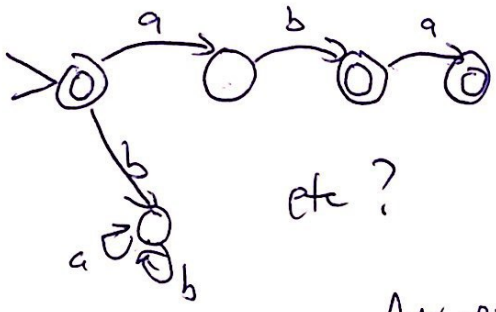
A configuration of an NFA is still a pair in $S \times \Sigma^*$ as for FSA

a string $w \in L(N)$ (is in the language defined by N)
iff $\exists p \in F \; (s_0, w) \vdash_N^* (p, \lambda)$

$(p, aw) \vdash_N (q, w)$ for $a \in \Sigma \cup \{\lambda\}$ if $(p, a, q) \in R$

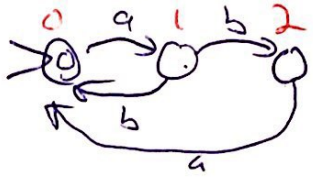
$(ab \vee aba)^*$

FSA:



etc ?

NFA N



An example non-deterministic string acceptance.

$(0, aba\,abababa) \vdash_N (1, baabababa)$

$(1, abababa) \dashv (0, aabababa)$

$(2, aabababa) \vdash (0, abababa)$

$(0, \lambda)$

$(2, a)$

$\dots \dashv (1, ba) \dashv (0, aba)$

$(1, bababa) \vdash \dots$

$(0, ababa)$

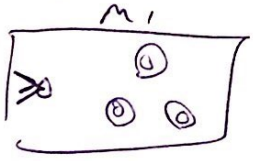
$(1, baba) \vdash \dots$

$(0, aba\,abababa) \in L(N)$

Thm Any language defined by a regex is definable by NFA

Pf. By induction on size of the regex.

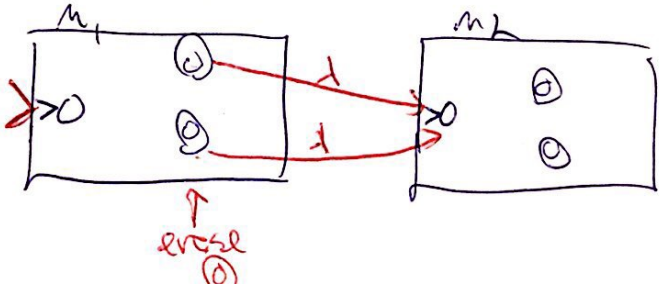
Base cases: regex is $\emptyset$, λ, or $a \in \Sigma$. easy to build NFA.



[note: every FSA is an NFA]

Ind cases Suppose NFA M_1 defines $L(\alpha_1)$
M_2 " $L(\alpha_2)$

- Build M defining $L(\alpha_1 \vee \alpha_2)$ see previous slide.
- Build M defining $L(\alpha_1 \alpha_2)$



Exercise Build M defining α_1^*

[Be sure to accept λ even if $\lambda \notin L(\alpha_1)$]