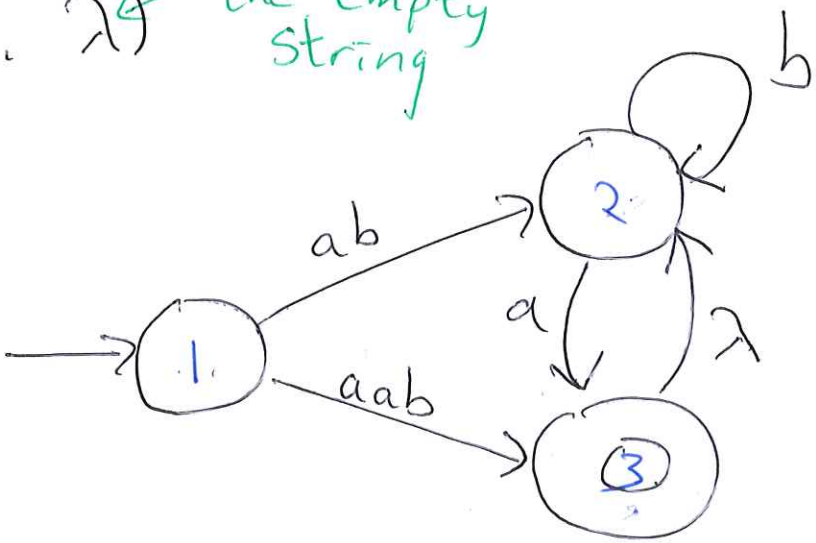
Nondeterministic Finite Automata

These are automata (like last week) where, given a state and a letter, there might be multiple ways of leaving that state.

I will (slightly differently from book) allow arrows to be labelled w/ any string (incl. λ ← the empty string)

E.g.



We are biased towards acceptance- if there is a path to ~~an~~ ~~accept~~ a final state the string is accepted.

One ~~of~~ way of thinking - we have a network of toll roads - chop off some part of the beginning of your string to pay toll - if you can pay out your entire

string and end up at a final state, you accept - if you can't, you reject.

It might ~~be~~ take a lot of work to tell a string should be rejected - b/c you have to make sure none of the possibilities work.

~~Hint~~ Is aab accepted? ✓

a ✗

ab ✗

aaba ✓

aabba ✓

abba ✓

abab ✗

Now I have to tell you ~~from~~ a formalization of how this works - which strings are accepted

We'll define a generalized transition function $\delta^*: Q \times \Gamma^* \rightarrow$ ~~set of states~~ $P(Q)$

↑ Power set of the set of states = set of all subsets of Q

~~Eg.~~ For our example, we should have

$\delta^*(1, a) = \emptyset = \{\}$

$\delta^*(1, aab) = \{2, 3\}$

$\delta^*(q, w) =$ ① take all ways of breaking up w into $w = uv$ (u & v are strings; could be empty)

② take $\delta^*(q, u)$ - this will give me a set of states

③ take all ways of pay a toll v from a state in S.

Mathematical formalization:

Data structure: A NFA (nondeterministic finite automaton) is a 5-tuple $M = (Q, \Gamma, \delta, q_0, F)$ where

Q is a finite set (of states)

Γ is a finite set (the alphabet)

δ is a finite relation: a finite subset of

$$(Q \times \Gamma^*) \times Q$$

(Γ^*: strings in alphabet Γ)

$q_0 \in Q$ the initial state

$F \subseteq Q$ the set of final states.

For our example:

$Q = \{1, 2, 3\}$

$\Gamma = \{a, b\}$

$q_0 = 1$

$F = \{3\}$

$$\delta = \{((1, ab), 2), ((1, aab), 3), ((2, a), 3), ((2, b), 2), ((3, \lambda), 2)\}$$

there exists

$$\delta^*(q,w) = \bigcup_{w=uv} \{ q'' \mid q' \in \delta^*(q,u) \text{ with } ((q',v),q'') \in \delta \}$$

$v \neq \lambda$

recursive case of def'n of δ^*

If $w = \lambda$, then

$\delta^*(q,w)$ = all states I can get to using free roads (i.e. paths labelled λ).

$\delta^*(q,\lambda)$ is recursively defined by

(B) $q \in \delta^*(q,\lambda)$

(R) If $((q,\lambda),q') \in \delta$, and $q \in \delta^*(q,\lambda)$, then $q' \in \delta^*(q,\lambda)$.

---

base case of def'n of δ^*.

Work this out for our example:

$\delta^*(1, aab)$ should be $\{2, 3\}$.

Is this what our def'n says?

~~Useful ways of s~~

Ways of splitting up $w = aab$ into $w = uv$

① $u = \lambda$ $v = aab$

② $u = a$ $v = ab$

③ $u = aa$ $v = b$

~~④ $u = aab$ $v = \lambda$~~

We should take the union over all these ways of splitting up w of

$\{q \mid$ there exists $q' \in \delta^*(q, u)$ with $((q', v), q) \in \delta\}$

① $\{q'' \mid$ there exists $q' \in \delta^*(1, \lambda)$ with $((q', aab), q'') \in \delta\} = \{3\}$ $\quad \delta^*(1, \lambda) = \{1\}$

② $\{q'' \mid$ there exists $q' \in \delta^*(1, a)$ with $((q', ab), q'') \in \delta\} = \{\}$