CSE596 Lecture Wed Sep 12 Fall 2018

To tell whether an NFA N accepts a string $x = x_1 \cdots x_n$ we want to compute, for each i, $\underline{0 \leq i \leq n}$, the set

$$R_i = \{ q \in Q : N \text{ can process } x_1 \cdots x_i \text{ from } s \text{ to } q \}$$

Goal: N accepts $x \iff R_n$ includes a state in F.

Origin: $R_0 = \{ q \in Q : N \text{ can process } \epsilon \text{ from } s \text{ to } q \}$
= the "epsilon closure" of $\{s\}$.

Helpful extra notation: not in any text(!) $\underline{\delta}(q, c) = \{ r :$ N can process c from q to r by first doing an arc on c then any optional trailing ϵ-arcs$\}$

Induction: Thus given R_{i-1}, we can calculate R_i by

$$R_i = \bigcup_{q \in R_{i-1}} \underline{\delta}(q, x_i).$$

This is correct (hand wave)

$\therefore$ **Theorem**: For every NFA $N = (Q, \Sigma, \delta, s, F)$ we can build an equiv^t DFA $M = (\mathcal{Q}, \Sigma, \Delta, S, \mathcal{F})$ by

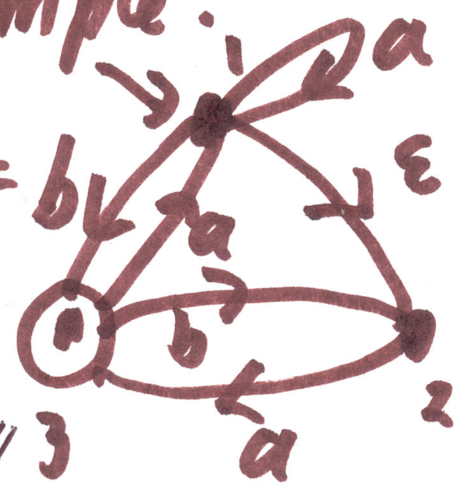
S = eps closure of $\{s\}$, i.e. R_0

$\mathcal{F} = \{ R \subseteq Q : R \cap F \neq \emptyset \}$

for all $P \subseteq Q$, $c \in \Sigma$, $\boxed{\Delta(P, c) = \bigcup_{p \in P} \underline{\delta}(p, c)}$

Example:

$\underline{\delta}(p,c)$: first c then εs.

N = 

{1, ε, 2} means 'whenever 1, then also 2!'

$\underline{\delta}(1,a) = \{1,2\}$ $\underline{\delta}(1,b) = \{3\}$
$\underline{\delta}(2,a) = \{3\}$ $\underline{\delta}(2,b) = \emptyset$
$\underline{\delta}(3,a) = \{1,2\}$ $\underline{\delta}(3,b) = \{2\}$.

$\Delta(P,c) = \bigcup_{p \in P} \underline{\delta}(p,c)$.

$S = \{1,\underline{2}\}$, not $\{1\}$

The DFA cannot have the states {1} or {1,3} because they have 1 but not 2.

Use Breadth First Search from S.

$\Delta(S,a) = \underline{\delta}(1,a) \cup \underline{\delta}(2,a) = \{1,2\} \cup \{3\} = \underbrace{\{1,2,3\}}_{\text{new state}}$

$\Delta(S,b) = \underline{\delta}(1,b) \cup \underline{\delta}(2,b) = \{3\} \cup \emptyset = \underline{\{3\}}$ also a new state

$\Delta(\{1,2,3\},a) = \{1,2\} \cup \{3\} \cup \{1,2\} = \{1,2,3\}$ again, not ne

$\Delta(\{1,2,3\},b) = \{3\} \cup \emptyset \cup \{2\} = \{2,3\}$, which $\underline{\text{is}}$ new.

$\Delta(\{3\},a) = \underline{\delta}(3,a) = \{1,2\}$

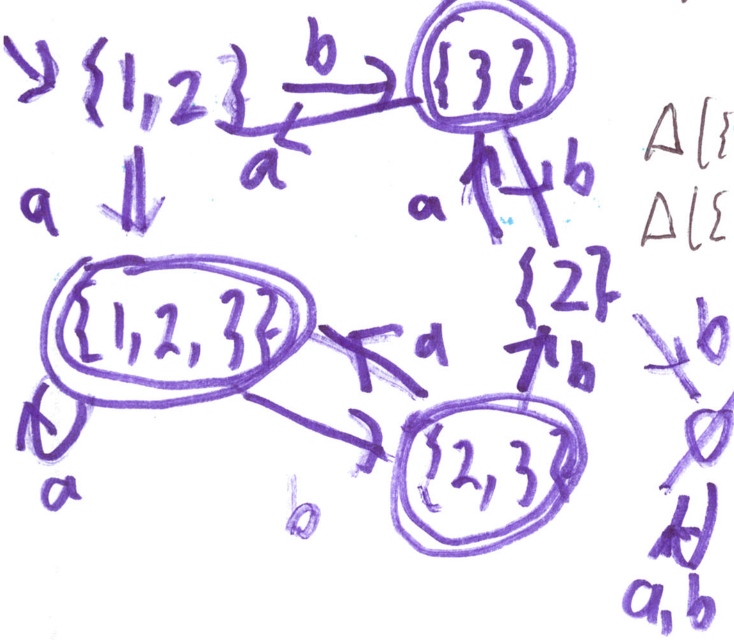
$\Delta(\{3\},b) = \underline{\delta}(3,b) = \{2\}$ new

$\Delta(\{2,3\},a) = \{3\} \cup \{1,2\} = \{1,2,3\}$

$\Delta(\{2,3\},b) = \emptyset \cup \{2\} = \{2\}$

$\Delta(\{2\},a) = \underline{\delta}(2,a) = \{3\}$ $\Delta(\emptyset,a) = \emptyset$

$\Delta(\{2\},b) = \underline{\delta}(2,b) = \emptyset$ $\Delta(\emptyset,b) = \emptyset$



No more new states: we say "The BFS has clos closed"

In lecture I pointed out:

- N can process "a" to any one of its three states. (Ditto aa, aaa, aba, etc, even $\underline{\text{bbaaa}}$.)
- N cannot process bbb from S at all. In M it goes to ∅.
- Indeed, N cannot process bbb at all: from {1,2,3} it goes to ∅. So {1,2,3} is not a continual accept condition, nor even continual "li

Example of an NFA that does have a continual-accept condition.

$L = \{x \in \{a,b\}^* : x \text{ has } aab \text{ in it}\}$

NFA:

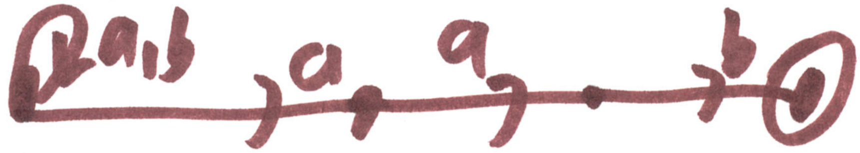
DFA has no dead state because no initial w can forbid having aab later. It has a continual-accept state once an aab occurs.

Regexp: $(a+b)^* aab (a+b)^*$.

$L' = \{x \in \{a,b\}^* : x \text{ } \underline{\text{ends}} \text{ in } aab\}$

Regexp $(a+b)^* aab$

NFA:



DFA has no dead state, no continual acc. state

$L'' = \{x \in \{a,b\}^* : x \text{ has an } aab \text{ } \underline{\text{and}} \text{ has } \underline{\text{no}} \text{ } bba\}$.

use cart prod on DFAs

DFA has dead state. Hard even to build NFA.