NFA $\equiv$ DFA

Recall: A nondeterministic finite automaton (NFA) is a machine

$$M = (Q, \Sigma, \Delta, Q_0, F)$$

ternary relation
$\Delta \subseteq Q \times \Sigma_\varepsilon \times Q$

set of initial states

Today: The power of nondeterminism

We said that every language in Reg is accepted by a DFA.

NFAs add the power of ε-transitions and choice.

Can they recognize languages outside Reg?

Thm: The class of languages recognized by NFAs is Reg

We show that every DFA has an equivalent NFA.

What is it?

We show that for any NFA, there is an equivalent DFA.

What does a correct guess boil down to, operationally?

Try all possibilities, choose the right one!

How can one simulate an NFA using a DFA?

An NFA can be in one of multiple different states on reading the same word!

Subset construction:

Each state of the DFA tracks some subset of states of the NFA

For now, consider NFAs without ε-transitions.

Consider an NFA $M = (Q, \Sigma, \Delta, Q_0, F)$

Want a DFA $M' = (Q', \Sigma, \delta', q_0', F')$ s.t. $L(M) = L(M')$.

Each state of M' needs to track (potentially) multiple states of M

So $Q' = 2^Q$. $\quad q_0' = Q_0 \subseteq Q$.

We now have to define δ'.

Consider a situation where $q_1, q_2, q_3 \in Q$, $a \in \Sigma$, and

$\Delta(q_1, a, q_1)$ and $\Delta(q_1, a, q_2)$.

What should $\delta'(\{q_1\}, a)$ be? $\{q_1, q_2\} \in Q'$

We first extend Δ to a function $\hat{\Delta} : 2^Q \times \Sigma^* \longrightarrow 2^Q$.

Any set of states of the NFA: $S \subseteq Q$

- stays the same on no input $\hat{\Delta}(S, \varepsilon) = \underline{S}$
- on a nonempty string, what? $\hat{\Delta}(\_, \_) = \_\_$

$$\hat{\Delta}(S, \varepsilon) = S$$

$$\hat{\Delta}(S, xa) = \{ q \mid \text{there is a } p \in \hat{\Delta}(S, x) \text{ s.t. } \Delta(p, a, q) \}$$

When does M accept s?

There is a run from some $q_0 \in Q_0$ to some $f \in F$ on s.

$$\hat{\Delta}(Q_0, s) \cap F \neq \emptyset$$

Lemma: For $S \subseteq Q$ and $x, y \in \Sigma^*$,

$$\hat{\Delta}(S, xy) = \hat{\Delta}(\hat{\Delta}(S, x), y)$$

Lemma: Suppose there exists an indexed set of subsets $S_i \subseteq Q$. Then, for any $x \in \Sigma^*$,

$$\hat{\Delta}\left(\bigcup_i S_i, x\right) = \bigcup_i \hat{\Delta}(S_i, x)$$

Exercise: Prove these lemmas

So back to our DFA. $M' = (Q', \Sigma, \delta', q_0, F')$

$Q' = 2^Q \qquad q_0 = Q_0 \subseteq Q$

$\delta'(q, a) = \hat{\Delta}(q, a) \qquad F' = \{ q \in Q' \mid q \cap F \neq \emptyset \}$

Lift δ' to $\hat{\delta} : Q' \times \Sigma^* \to Q' \quad \left(\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a) \right)$

Thm: For any $S \subseteq Q$ and $w \in \Sigma^*$,

$$\hat{\delta}(S, w) = \hat{\Delta}(S, w).$$

Thm: $\mathcal{L}(M) = \mathcal{L}(M')$ — consider which strings are accepted and what those conditions are

Exercise: Prove these statements

How do we handle ε-transitions?

Thm: If $\mathcal{L}$ is accepted by an NFA with ε-transitions, it is accepted by an NFA without any ε-transitions

Suppose $\mathcal{L} = \mathcal{L}(M)$ where $M = (Q, \Sigma_\varepsilon, \Delta, Q_0, F)$.

Define ε-closure: $Q \to 2^Q$ as follows:

$\varepsilon\text{-closure}(q) = \{ q' \mid$ there is a path from q to q' in M consisting only of ε-transitions $\}$

Now, $\hat{\Delta}(q, \varepsilon) = \varepsilon\text{-closure}(q)$

$\hat{\Delta}(q, xa) = \{ p \mid$ there is some $r \in \hat{\Delta}(q, x)$ s.t. $\Delta(r, a, p) \}$

Define a new NFA

$M' = (Q', \Sigma, \Delta', Q_0', F')$, where

$Q' = Q$ and $Q_0' = Q_0$

For any $a \in \Sigma$, $\Delta'(q, a, q')$ holds iff $q' \in \mathcal{E}\text{-closure}(\hat{\Delta}(q, a))$

$F' = F \cup \{ q \in Q_0 \mid \mathcal{E}\text{-closure}(q) \cap F \neq \emptyset \}$

Thm: $\mathcal{L}(M) = \mathcal{L}(M')$

The proof proceeds by induction on the length of an input word.

What is the base case? Finish this proof.

When we move from an NFA to a DFA, there is potentially an exponential blow-up in the number of states.

Consider, over $\Sigma = \{a, b\}$, the following language

$$\mathcal{L} = \{ s \mid s \text{ contains aba or aa as a substring} \}$$

What is a finite automaton recognizing $\mathcal{L}$?

But what if I have to ask my text editor to find all such words?

Is there a less informal way to do it? Regular expressions

What are the basic regular expressions?

- a, for every $a \in \Sigma$
- ε
- ϕ

Suppose φ and ψ are regular expressions. Then, so are:

- $\varphi \cdot \psi$
- $\varphi + \psi$
- φ^*
- $\sim \varphi$

$$\mathcal{L} = \{ s \mid s \text{ contains } aba \text{ or } aa \text{ as a substring} \}$$

What is a regular expression that represents $\mathcal{L}$?