Lecture 26: NFA-recognizable = DFA-recognizable

- We show that NFA-recognizable languages are DFA-recognizable and vice versa.
- We introduce ε-NFA, which also add no new power.

Applications

- This is an example of adding "language features" by "compiling" them to the basic language.
- Although NFA are not more powerful than DFA, for more general computations the question of whether nondeterminism can be removed is open (e.g., factoring is easy to solve non-deterministically).
Claim: \( L \) is DFA-recognizable iff \( L \) is NFA-recognizable.

Proof:

(\( \Rightarrow \)) \( \text{wfs of } L \text{ is DFA-recognizable} \implies L \text{ is NFA-recognizable.} \)

Let \( M \) be a DFA that recognizes \( L \). Well, \( M \) is also an NFA, so \( L \) is NFA-recognizable.

(\( \Leftarrow \)) \( \text{wfs of } L \text{ is NFA-recognizable} \Rightarrow L \text{ is DFA-recognizable.} \)

Assume \( \exists \) an NFA \( N \) with \( L = L(N) \).

We want to build a DFA \( M \) with \( L(M) = L(N) \).

Idea: Each state of \( M \) will represent a set of states of \( N \).
Let \( Q = 2^Q \), \( \Sigma = \Sigma \), \( Q^* = Q \)
\[ \delta : Q^* \times \Sigma \to Q \]
\[ \delta : 2^Q \times \Sigma \to 2^Q \]
\[ \delta(M, x) = S \]
\[ \delta(N, x, y) = S \]
\[ \delta(M, x) = \delta(N, x, y) \]
\[ \delta(M, x) = \delta(N, x, y) \]

Proof of subclaim:

By induction on \( x \), let \( P(x) \):

\[ P(x) \] is defined as:

For \( P(x) \), assume \( P(x) \), plug in \( \hat{\delta}_M, \hat{\delta}_N, \delta_M, \delta_N \), \( \delta_M, \delta_N \).

Well:

\[ \hat{\delta}_M(q_0, x) = \delta_M(\hat{\delta}_M(q_0, x), a) \]
\[ = \delta_M(\hat{\delta}_N(q_0, x), a) \]
\[ = \ldots \]

Claim: \( L(M) = L(N) \)

Subclaim (below):

Define of \( L(M) \)

Define of \( L(N) \)

Define of \( A_M \)

Define of \( A_N \)
Definition: An ε-NFA $N$ is like an NFA, but we allow "ε-transitions".

can follow ε-transitions for free (without processing any characters).