Oberlin College
CSCI 383
Fall 2026
Lecture 5. Nondeterministic Finite Automata
Kozen - Section 5

Last Time: What languages are regular? Closure properties.

Today: How important are the details of our computational model?

In particular, too many/too few arrows.

Violating example: Draw NFA for $\{x \in \{0,1\}^* \mid \text{the fourth symbol from the right is a } 1\}$.

- There are two transitions for 1 out of first state.
- The last state doesn't have a transition.

Nondeterministic Finite Automata

Idea: Guess which path to take.

Informally: An NFA N accepts a string x if there is *some* way of reading x and taking appropriate transitions in N that ends in an accept state.

Notes:

- If you hit a dead end, you fail.
- Not all guesses must lead to an accept state.

Example: 11011010. When should I guess to go right?

Example: 11000010. When should I guess to go right?

Some more examples: (If needed.)

As with DFAs, $L(N)$ denotes the set of strings accepted by N.

What's an English description of $L(N)$?

The process of guessing which transition to take is called *nondeterminism*.

NP stands for *nondeterministic polynomial time*. By the end of the semester, you should be able to figure out exactly why.

Another Example: $\{x \in \{a,b\} \mid \text{every } b \text{ is preceded by an } a\}$.

- *aaaab*?
- *aaabba*?
- *bbbb*?

What's the language this NFA accepts?

Formal Definition

An NFA is a 5-touple $N = (Q, \Sigma, \Delta, S, F)$, where:

- Q - states (same as before) (In example: $\{1, 2\}$)
- Σ - finite alphabet (same as before) (In example: $\{a, b\}$)
- Δ - transition function

  (What did our transition functions take as input before? What was the output?)

  $\Delta : Q \times \Sigma \to 2^Q$ where:

  - $Q \times \Sigma$ = the set of all (state, symbol) pairs
  - 2^Q = set of all subsets of Q

  (In this course, as much as possible, we'll try to use capital letters for sets of things, and lower case for non-sets.)
  Intuition: $\Delta(q, c)$= Where can I get in one step from q on reading c?

  - $\Delta(1, a) = \{1, 2\}$
  - $\Delta(2, b) = \{1\}$
  - $\Delta(1, b) = \emptyset$
    (That's right, $\Delta(q, c)$ can be empty! We can have "dead ends.")
- S - set of start states.

  You get to pick where to start.

  (Add a new start state that adds strings that start with *bb*.)

  (What's S here?)

  (What's $L(N)$?)
- F - the set of final states. (In example: $\{1\}$.)

Work out first example.

Any DFA can be turned into an NFA: have Δ output singleton sets, $S = \{s\}$.

Defining Acceptance.

First, we must define inductive extension $\hat{\Delta}$.

Intuitively: Starting from some state $q \in A$, what states can we reach after reading x?

Example from first automaton.

Example: In first NFA today, what is:

- $\hat{\Delta}(\{a\}, 0)$?
- $\hat{\Delta}(\{a\}, 01)$?
- $\hat{\Delta}(\{a\}, 010)$?

Formally:

- $\hat{\Delta}(A, \epsilon) = A$
- $\hat{\Delta}(A, xa) = \bigcup_{q \in \hat{\Delta}(A,x)} \{\Delta(q, a)\}$

Def: An NFA *accepts* a string x if $F \cap \hat{\Delta}(S, x) \neq \emptyset$.

i.e. if you look at the places you can get from the start states, there's at least one final state in there.

All regular languages are accepted by an NFA.

Question: Are there non-regular languages accepted by some NFA?