COMPSCI 501: Formal Language Theory

Lecture 3: Nondeterministic Finite Automata

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

January 28, 2019

Nondeterminism

Deterministic: next state uniquely determined by state and input
Nondeterministic: several choices (incl. zero) at any point
computation tree

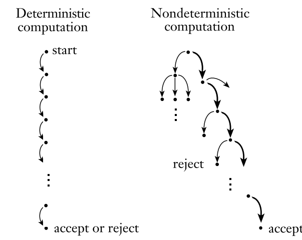
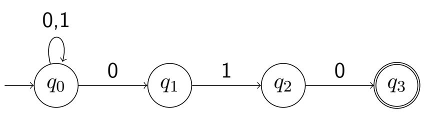


FIGURE 1.28
Deterministic and nondeterministic computations with an accepting branch

Think of it as forking multiple computations in *parallel*

Example: Nondeterministic Finite Automaton

Example: accept binary strings ending in 010



State q_0 has **two** transitions on input 0.
State q_1 has **no** transition on input 0
run on input string might "get stuck" $\Rightarrow$ reject

String is accepted if there is **some** run ending in accepting state.
Any string accepted must end in 010.
Any string ending in 010 is accepted: NFA can stay in initial state until the last 3 symbols ("guesses" when to move).

NFA Definition

A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of *states*
- Σ is the finite *alphabet* (of input *symbols*)
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the *transition function*

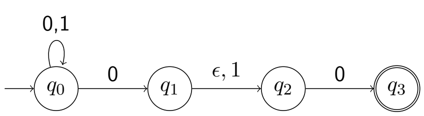
alternatively: $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*

- $q_0 \in Q$ is the *start state*
- $F \subseteq Q$ is the *set of accept states*.

Any relation $R \subseteq A \times B$ corresponds to a function $f_R : A \to \mathcal{P}(B)$
$f_R(x) = \{y \in B \mid (x, y) \in R\}$.

ϵ-transitions

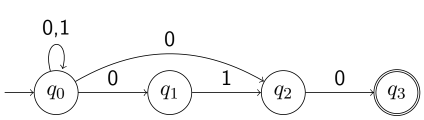
Transition marked with ϵ (empty string) is taken *without consuming* input symbol.



Denote $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$. Revised definition:

- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$

Textbook directly defines NFAs as allowing ϵ-transitions.
Other sources distinguish NFAs and ϵ-NFAs.
Any ϵ-NFAs can be transformed into an equivalent NFA without ϵ.



Equivalence of NFAs and DFAs

Subset construction: consider the **set** of states that the automaton could be in at any given time

Construct DFA $M = (Q', \Sigma, \delta', q_0', F')$

- $Q' = \mathcal{P}(Q)$ a state of M is a set of states of N
- $\delta'(R, a) = \cup_{r \in R} \delta(r, a)$ for $R \in Q'$ $(R \subseteq Q)$
- $q_0' = \{q_0\}$ (set with the one initial state)
- $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$ (contains some accept state)

Subset construction: with ϵ-transitions

If NFA is in any given state, it could also be in any state reachable by one or more ϵ-transitions

Define $E(R) = \{q \mid q \text{ reachable from } R \text{ along } \epsilon\text{-transitions}\}$

Transition relation changes to

- $\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}$

Subset construction

If NFA has k states, equivalent DFA could have up to 2^k states.

Can you give an example?

What can we say the resulting DFA has $\emptyset$ as state?

There is some w such that no string with prefix w is accepted.

Regular languages, again

We've seen any NFA can be converted to an equivalent DFA.

Corollary: A language is regular if and only if some *nondeterministic* finite automaton recognizes it.

$\Rightarrow$ If the language is accepted by an NFA,
and the NFA has an equivalent DFA
then the language is accepted by a DFA, thus regular.

$\Leftarrow$ If the language is regular, it's accepted by a DFA.
Any DFA is also a NFA.

Exercise: All-NFA

An **all-NFA** is like an NFA, except that is accepts a string $x \in \Sigma^*$ if *every* possible state of the NFA after reading x is accepting.

Is the class of languages accepted by an all-NFA still the class of regular languages?

Closure under Union

Add new initial state with ϵ-transitions to both initial states

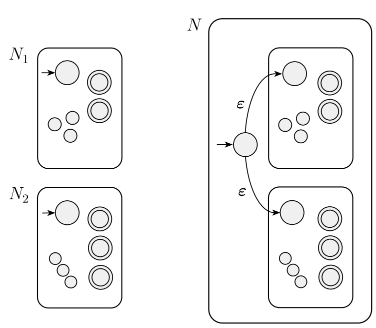
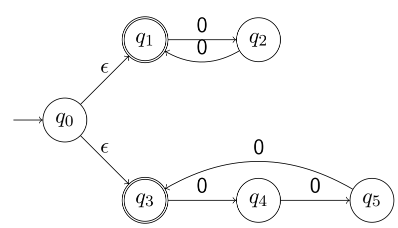


FIGURE 1.46
Construction of an NFA N to recognize $A_1 \cup A_2$

Example: strings with certain lengths

Consider a one-letter alphabet $\Sigma = \{0\}$

Strings which have length multiple of 2 *or* multiple of 3



Closure under Concatenation

Add ϵ-transitions from all accept states of N_1 to initial state of N_2

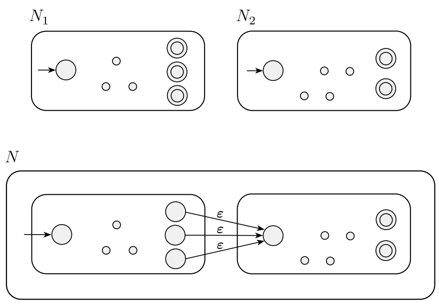


FIGURE 1.48
Construction of N to recognize $A_1 \circ A_2$

Closure under Kleene Star

Add ϵ-transitions from all accept states to initial state, and new initial (and accepting) state with ϵ-transitions to original one.

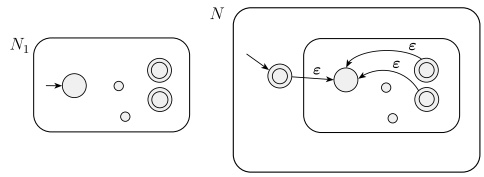


FIGURE 1.50
Construction of N to recognize A^*

Do we need the extra initial state?
What if we make the original initial state accepting?

Can we construct these without ϵ-transitions?

Yes, since we can convert any ϵ-NFA into one without ϵ:

Union: new initial state, add transitions of *both* initial states

Concatenation: transitions from all accepting states of N_1 to *successors* of initial state of N_2
(including from initial state of N_1, if accepting)
if $\epsilon \in L(N_1)L(N_2)$, initial state stays accepting

Star: initial state becomes accepting
transitions from any accepting state to corresponding successors of initial state