Lecture 3:
• Nondeterminism
• Equivalence of NFAs and DFAs
• More on closure properties

Reading:
Sipser Ch 1.1-1.2

Mark Bun
January 29, 2020
A Nondeterministic Finite Automaton (NFA) accepts if there is a way to make it reach an accept state.
**Example:** Does this NFA accept the string 1100?
Some special transitions

- Transition on 0
- Transition on 1
- Transition on ε
- Transition on 1
Example

$L(M) = \varepsilon$
Example

\[ L(M) = \]

$0,1$

\[ \text{Example} \]

$L(M) =$
Formal Definition of a NFA

An NFA is a 5-tuple \( M = (Q, \Sigma, \delta, q_0, F) \)

- \( Q \) is the set of states
- \( \Sigma \) is the alphabet
- \( \delta : Q \times \Sigma \varepsilon \rightarrow P(Q) \) is the transition function
- \( q_0 \in Q \) is the start state
- \( F \subseteq Q \) is the set of accept states

\( M \) accepts a string \( w \) if there exists a path from \( q_0 \) to an accept state that can be followed by reading \( w \).
Example

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

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$F = \{q_4\}$

$\delta(q_2, 1) = \ $

$\delta(q_3, 1) = \ $
Example

$N = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$F = \{q_3\}$

$\delta(q_0, 0) = \cdots$

$\delta(q_0, 1) = \cdots$

$\delta(q_1, \varepsilon) = \cdots$

$\delta(q_2, 0) = \cdots$
Nondeterminism

Ways to think about nondeterminism

- (restricted) parallel computation
- tree of possible computations
- guessing and verifying the “right” choice
Why study NFAs?

• Not really a realistic model of computation: Real computing devices can’t really try many possibilities in parallel

But:

• Useful tool for understanding power of DFAs/regular languages
• NFAs can be simpler than DFAs
• Lets us study “nondeterminism” as a resource (cf. P vs. NP)
NFAs can be simpler than DFAs

A DFA that recognizes the language \( \{1\} \):

An NFA that recognizes the language \( \{1\} \):
Sometimes DFAs **must** be larger

**Theorem.** Every DFA for the language \{1\} must have at least 3 states.

**Proof:**
Equivalence of NFAs and DFAs
Equivalence of NFAs and DFAs

Every DFA is an NFA, so NFAs are at least as powerful as DFAs

Theorem: For every NFA $N$, there is a DFA $M$ such that $L(M) = L(N)$

Corollary: A language is regular if and only if it is recognized by an NFA
Equivalence of NFAs and DFAs (Proof)

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA

Goal: Construct DFA $M = (Q', \Sigma, \delta', q_0', F')$ recognizing $L(N)$

**Intuition:** Run all threads of $N$ in parallel, maintaining the set of states where all threads are.

**Formally:** $Q' = P(Q)$

“The Subset Construction”
NFA -> DFA Example

\[ \text{a} \xrightarrow{1} \text{b} \]
Subset Construction (Formally)

Input: NFA \( N = (Q, \Sigma, \delta, q_0, F) \)

Output: DFA \( M = (Q', \Sigma, \delta', q_0', F') \)

\( Q' \)

\( \delta' : Q' \times \Sigma \rightarrow Q' \)

\( \delta'(R, \sigma) = \)

for all \( R \subseteq Q \) and \( \sigma \in \Sigma \).

\( q_0' = \)

\( F' = \)
NFA -> DFA Example
Subset Construction (Formally)

Input: NFA $N = (Q, \Sigma, \delta, q_0, F)$
Output: DFA $M = (Q', \Sigma, \delta', q_0', F')$

$Q' = P(Q)$

$\delta' : Q' \times \Sigma \rightarrow Q'$

$\delta'(R, \sigma) = \bigcup_{r \in R} \delta(r, \sigma)$ for all $R \subseteq Q$ and $\sigma \in \Sigma$.

$q_0' = \{q_0\}$

$F' = \{R \in Q' \mid R$ contains some accept state of $N\}$
Proving the Construction Works

Claim: For every string \( w \), running \( M \) on \( w \) leads to state

\[ \{ q \in Q | \text{There exists a computation of } N \text{ on input } w \text{ ending at } q \} \]

Proof idea: By induction on \( |w| \)
Historical Note

Subset Construction introduced in Rabin & Scott’s 1959 paper “Finite Automata and their Decision Problems”

1976 ACM Turing Award citation

For their joint paper "Finite Automata and Their Decision Problem," which introduced the idea of nondeterministic machines, which has proved to be an enormously valuable concept. Their (Scott & Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.
Is this construction the best we can do?

Subset construction converts an \( n \) state NFA into a \( 2^n \)-state DFA.

Could there be a construction that always produces, say, an \( n^2 \)-state DFA?

**Theorem:** For every \( n \geq 1 \), there is a language \( L_n \) such that

1. There is an \((n + 1)\)-state NFA recognizing \( L_n \).
2. There is no DFA recognizing \( L_n \) with fewer than \( 2^n \) states.

**Conclusion:** For finite automata, nondeterminism provides an exponential savings over determinism (in the worst case).
More on Closure Properties
Operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

**Union:** $A \cup B$

**Concatenation:** $A \circ B = \{ab \mid a \in A, b \in B\}$

**Star:** $A^* = \{a_1 a_2 \ldots a_n \mid n \geq 0 \text{ and } a_i \in A\}$

**Complement:** $\overline{A}$

**Intersection:** $A \cap B$

**Reverse:** $A^R = \{a_1 a_2 \ldots a_n \mid a_n \ldots a_1 \in A\}$

**Theorem:** The class of regular languages is closed under all six of these operations.
Theorem. The reverse of a regular language is also regular

Proof: Let $L$ be a regular language and $M$ be a DFA recognizing it. Construct an NFA $M'$ recognizing $L^R$:

- Define $M'$ as $M$ with the arrows reversed.
- Make the start state of $M$ be the accept state in $M'$.
- Make a new start state that goes to all accept states of $M$ by $\varepsilon$-transitions.
Closure under Concatenation

Concatenation: \( A \circ B = \{ ab \mid a \in A \text{ and } b \in B \} \)

**Theorem.** If \( A \) and \( B \) are regular, \( A \circ B \) is also regular.

**Proof:** Given DFAs \( M_A \) and \( M_B \), construct NFA by

1. Connecting all accept states in \( M_A \) to the start state in \( M_B \).
2. Make all states in \( M_A \) non-accepting.

\[ L(M_A) = A \quad \Rightarrow \quad L(M_B) = B \]
A Mystery Construction

Given DFAs $M_A$ recognizing $A$ and $M_B$ recognizing $B$, what does the following NFA recognize?
Closure under Star

Star: \( A^* = \{ a_1 a_2 \ldots a_n | n \geq 0 \text{ and } a_i \in A \} \)

**Theorem.** If \( A \) is regular, \( A^* \) is also regular.