Automata Theory and Formal Grammars: Lecture 2
Deterministic and Nondeterministic Finite Automata

Last Time
- Sets Theory (Review?)
- Logic, Proofs (Review?)
- Words, and operations on them: $w_1 \circ w_2, w^i, w^*, w^+$
- Languages, and operations on them: $L_1 \circ L_2, L^i, L^*, L^+$

Today
- Deterministic Finite Automata (DFAs) and their languages
- Closure properties of DFA languages (the product construction)
- Nondeterministic Finite Automata (NFAs) and their languages
- Relating DFAs and NFAs (the subset construction)

Fibonacci as a Recursively Defined Set

The $n^{th}$ Fibonacci number $f(n)$:

- $f(0) = 0$
- $f(1) = 1$
- $f(n) = f(n-1) + f(n-2)$, for $n \geq 2$

As a recursively defined set (relation)

\[
\begin{align*}
F_0 &= \emptyset \\
F_{i+1} &= \{ \langle 0, 0 \rangle, \langle 1, 1 \rangle \} \\
&\quad \cup \left\{ \langle n, f_{n_1} + f_{n_2} \rangle \middle| \begin{array}{l}
\langle n_1, f_{n_1} \rangle \in F_i \quad \text{and} \\
\langle n_2, f_{n_2} \rangle \in F_i \\
n = n_1 + 1 = n_2 + 2
\end{array} \right\}
\end{align*}
\]

For example:

\[
\begin{align*}
F_0 &= \emptyset \\
F_1 &= \{ \langle 0, 0 \rangle, \langle 1, 1 \rangle \} \\
F_2 &= \{ \langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \\
F_3 &= \{ \langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 1 \rangle, \langle 3, 2 \rangle \} \\
F_4 &= \{ \langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 3 \rangle \} \\
F_5 &= \{ \langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 3 \rangle, \langle 5, 5 \rangle \}
\end{align*}
\]
Conventions

- $\Sigma$ is an arbitrary alphabet. (In examples, $\Sigma$ should be clear from context.)
- The variables $a$–$e$ range over letters in $\Sigma$.
- The variables $u$–$z$ range over words over $\Sigma^*$.
- The variables $p$–$q$ range over states in $Q$.

Recall

For any string $w$ and language $L$:

\[ w \circ \varepsilon = w = \varepsilon \circ w \]  

(1)

\[ L \circ \{\varepsilon\} = L = \{\varepsilon\} \circ L \]  

(2)

\[ L^* = \{\varepsilon\} \cup L \circ L^* \]  

(3)

$L^*$ is closed with respect to concatenation, for any $L$:

if $u \in L^*$ and $v \in L^*$ then $u \circ v \in L^*$

Finite Automata

... are “machines” for recognizing languages!
- They process input words a symbol at a time.
- An “accept light” flashes if the symbols read in so far are “OK”.

Formal Definition of Finite Automata

A finite automaton (DFA) is a quintuple $\langle Q, \Sigma, q_0, \delta, A \rangle$ where:
- $Q$ is a finite non-empty set of states;
- $\Sigma$ is an alphabet;
- $q_0 \in Q$ is the start state;
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function; and
- $A \subseteq Q$ is the set of accepting (final) states.
DFA Acceptance

Given a DFA \( M = (Q, \Sigma, q_0, \delta, A) \) and word \( w \in \Sigma^* \):

- \( M \) should accept \( w \) if in processing \( w \) a symbol at a time, \( M \) goes to an accepting state.
- To formalize this we define a function \( \delta^* : Q \times \Sigma^* \rightarrow Q \)

\[ \delta^*(q, w) \] should be the state reached from \( q \) after processing \( w \).

- How to define \( \delta^* \)?

Example of \( \delta^* \)

![Diagram of DFA with states and transitions]

\[ \begin{align*}
\delta^*(0, aab) &= \delta^*(\delta(0, a), ab) = \delta^*(2, ab) \\
&= \delta^*(\delta(2, a), b) = \delta^*(3, b) \\
&= \delta^*(\delta(3, b), \varepsilon) = \delta^*(1, \varepsilon) \\
&= 1
\end{align*} \]

What is \( \delta^*(0, abaa) \)?

Language of a Finite Automaton

A DFA accepts a word if it reaches an accepting state after "consuming" the word.

**Definition** Let \( M = (Q, \Sigma, q_0, \delta, A) \) be a DFA. Then \( \delta^* : Q \times \Sigma^* \rightarrow Q \) is defined recursively:

\[ \delta^*(q, w) = \begin{cases} 
q & \text{if } w = \varepsilon \\
\delta^*(\delta(q, a), w') & \text{if } w = aw' \text{ and } a \in \Sigma 
\end{cases} \]

\( \delta^*(q, w) = q' \) if \( q' \) the state reached by processing \( w \), starting from \( q \).
Definition: A language $L \subseteq \Sigma^*$ is a DFA language if there exists a DFA $M$ such that $L = L(M)$.

- Is the set of Java numeric constants a DFA language?
  
  $0\times E, 15, 017, 15L, 15.0, 1.5e1, 1.5E1$

  Yes! To show it build a DFA.

- Is the set of strings of balanced parentheses a DFA language?
  
  $\varepsilon, ab, aabb, aaabbb,$

  No! To show it ... attend lecture 4.
Closed Sets

Let $f$ be a unary operation $f : U \to U$. A subset $S \subseteq U$ is closed under $f$ — equivalently, $f$ preserves $S$ — if

$$\forall s \in S. f(s) \in S$$

Let $g$ be a binary operation $g : U \times U \to U$. A subset $S \subseteq U$ is closed under $g$ — equivalently, $g$ preserves $S$ — if

$$\forall \langle s_1, s_2 \rangle \in S \times S. f(s_1, s_2) \in S$$

- Naturals are closed under addition, not subtraction.
- Integers are closed under multiplication, not division.
- Rationals are closed under division, not square root.
- Reals are closed under square root, not exponentiation.
- Complex are closed under exponentiation.

Closure Properties for DFA Languages

- We would like to see what operations on languages “preserve” the property of being recognizable by a DFA.
- For example, suppose we wish to show the following:
  Let $L_1$ and $L_2$ be DFA languages. Then $\overline{L_1}$ and $L_1 \cap L_2$ are also DFA languages.
- How do we prove this? Via constructions.

Complementation

**Theorem** Let $L \subseteq \Sigma^*$ be a DFA language. Then so is $\overline{L}$.

Since $L$ is a DFA language we know there is a DFA $M$ accepting it. How can we build a DFA for $\overline{L}$?

**Idea** Reverse the accepting and nonaccepting states in $M$!

The proof formalizes this idea.

Proof

- Suppose $L$ is a DFA language. By definition, there must be a DFA $M$ such that $L(M) = L$.
- Fix $M = \langle Q, \Sigma, q_0, \delta, A \rangle$.
- Let $\overline{M} = \langle Q, \Sigma, q_0, \delta, Q - A \rangle$. We show that $L(\overline{M}) = \overline{L}$.
  - For any $w \in \Sigma^*$,
    $$\delta^*(q_0, w) \notin A \iff \delta^*(q_0, w) \in Q - A$$
  - This holds trivially by induction on length of $w$.
  - Thus, for any $w, w' \in L(\overline{M})$ if and only if $w \notin L(M)$.
  - Thus, $L(\overline{M}) = \overline{L}$. QED
Intersection

**Theorem** Let \( L_1, L_2 \subseteq \Sigma^* \) be DFA languages. Then \( L_1 \cap L_2 \) is a DFA language.

To prove this we will use the **Product Construction**.

- Given two DFAs \( M \) and \( N \), the product construction builds a new DFA \( \Pi(M, N) \) that “runs” \( M \) and \( N \) in parallel.
- \( \Pi(M, N) \) then accepts a word iff both \( M \) and \( N \) do.
- So \( L(\Pi(M, N)) = L(M) \cap L(N) \).

**How do we define \( \Pi \)?**

**The Product Construction**

Let \( M = \langle Q_M, \Sigma, q_M, \delta_M, A_M \rangle \) be a DFA.

Let \( N = \langle Q_N, \Sigma, q_N, \delta_N, A_N \rangle \) be a DFA.

Define \( \Pi(M, N) \) as

\[
\Pi(M, N) = \langle Q_M \times Q_N, \Sigma, \langle q_M, q_N \rangle, \delta_{MN}, A_M \times A_N \rangle
\]

where

\[
\delta_{MN}(\langle q_1, q_2 \rangle, a) = \langle \delta_M(q_1, a), \delta_N(q_2, a) \rangle
\]

**Lemma** For any \( w \in \Sigma^*, q_1 \in Q_M \), and \( q_2 \in Q_N \),

\[
\delta^*_{MN}(\langle q_1, q_2 \rangle, w) = \langle \delta^*_M(q_1, w), \delta^*_N(q_2, w) \rangle.
\]

**Proof?**

And how does this help show that \( L(\Pi(M, N)) = L(M) \cap L(N) \)?
A Corollary about Closure for DFA Languages

What’s a “corollary”? An “obvious consequence”.

**Corollary**

Let \( L_1, L_2 \subseteq \Sigma^* \) be DFA languages. Then so are \( L_1 \cup L_2 \) and \( L_1 - L_2 \).

Why is this an “obvious consequence” of what we have seen before?

---

Nondeterministic Finite Automata

- Regular DFAs require exactly one transition per state for each input symbol.
- **Nondeterministic** FAs allow any number of transitions!

Why study NFAs? Because they are easier to work with sometimes....

---

NFAs Can Be Smaller Than DFAs!

Consider language \( L \subseteq \{0, 1\}^* \) given by regular expression \((0 + 1)^*0(0 + 1)(0 + 1)(3^{rd} \text{ symbol from right is a } 0)\).
A nondeterministic finite automaton (NFA) is a quintuple $\langle Q, \Sigma, q_0, \delta, A \rangle$ where:

- $Q$ is a finite set of states;
- $\Sigma$ is the input alphabet;
- $q_0 \in Q$ is the start state;
- $A \subseteq Q$ is the set of accepting states; and
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function.

Idea: $\delta(q, a)$ records the set of states reachable from $q$ via an $a$-transition.

To formalize acceptance we first define a function $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$; $\delta^*(q, w)$ contains all the states reachable from $q$ after processing $w$.

Definition: Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a NFA. Then $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$ is defined as follows.

\[
\delta^*(q, w) = \begin{cases} 
\{q\} & \text{if } w = \varepsilon \\
\bigcup_{q' \in \delta(q, a)} \delta^*(q', w') & \text{if } w = aw' \text{ and } a \in \Sigma
\end{cases}
\]

Note that $\delta^*(q, w)$ gives us the set of all possible outcomes of processing $w$ from state $q$.

As with DFAs, the language of a NFA consists of the words that it accepts.

In a NFA nondeterministic choices require “guessing”: which transition should be taken? Some paths may lead to accepting states, whereas others do not.

A NFA accepts a word if it is possible to make the guesses so that we reach an accepting state.

This is called angelic nondeterminism. We have access to an oracle that always guesses correctly.
Languages of NFAs: Formal Definition

**Definition** Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a NFA, and let $w \in \Sigma^*$.

- $M$ accepts $w$ if $\delta^*(q_0, w) \cap A \neq \emptyset$.
- The language, $L(M)$, of $M$ is defined by: $L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$

So $M$ accepts $w$ if it is possible to reach an accepting state after processing $w$.

So What Is Relationship Between DFAs and NFAs?

**Theorem**

1. For any DFA $M$ there is a NFA $N$ such that $L(N) = L(M)$.
2. For any NFA $N$ there is a DFA $M$ such that $L(M) = L(N)$.

Proof of 1 is easy, since any DFA “is” a NFA. But 2?

- Idea behind proof is to define DFA that “tracks” behavior of NFA on a given input word.
- This construction is often called the subset construction because states in the DFA correspond to set of states in the NFA.

The Subset Construction: Intuition

The Subset Construction

Let $N = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a NFA.
We want to construct a DFA $D(N)$ accepting the same language.
States in $D(N)$ will be sets of states from $N$.
Let $P$ range over states of $D(N)$.
$P \in 2^Q$, that is, $P \subseteq Q$.

$$D(N) = \langle 2^Q, \Sigma, \{q_0\}, \delta_{DN}, A_{DN} \rangle$$

where

$$\delta_{DN}(P, a) = \bigcup_{q \in P} \delta(q, a)$$

$$A_{DN} = \{ P \mid P \in 2^Q \text{ and } P \cap A \neq \emptyset \}$$

Note that $\delta^*(q_0, w) \in Q$, whereas $\delta_{DN}(\{q_0\}, w) \subseteq Q$. 
Example of Subset Construction

Note. Only reachable states in \( D(N) \) are represented. (In practice, not all subsets of \( Q \) are reachable from \( \{q_0\} \), and these need not be added explicitly to \( D(N) \).)

Correctness of Subset Construction

Let \( N = \langle Q, \Sigma, q_0, \delta, A \rangle \) be a NFA.

\[
D(N) = \langle 2^Q, \Sigma, \{q_0\}, \delta_{DN}, A_{DN} \rangle
\]

where — for \( P \in 2^Q \) —

\[
\delta_{DN}(P, a) = \bigcup_{q \in P} \delta(q, a)
\]

and

\[
A_{DN} = \{ P \mid P \cap A \neq \emptyset \}.
\]

**Theorem** For any NFA \( N, L(N) = L(D(N)) \).

Recall that \( \delta^*(q_0, w) \in Q \), whereas \( \delta^*_{DN}(\{q_0\}, w) \subseteq Q \).

The proof relies on the following observations. For any \( w \in \Sigma^* \):

- \( \delta^*(q_0, w) = \delta^*_{DN}(\{q_0\}, w) \)
- \( \delta^*(q_0, w) \cap A \neq \emptyset \) if and only if \( \delta^*_{DN}(\{q_0\}, w) \in A_{DN} \)

Consequently, \( w \in L(N) \) if and only if \( w \in L(D(N)) \).